Altímetro barométrico assistido por modelos numéricos de previsão meteorológica

José Ricardo Alves Fonseca
Mestrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2016

Orientador
Sérgio Crisóstomo, Professor Auxiliar, Faculdade de Ciências da Universidade do Porto

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, / /

Abstract

The inclusion of several sensors in smartphones combined with the Internet access capacity made it possible to develop applications with increasing monitoring capabilities and activities on the surrounding environment.

The barometric sensor, presented in models of some smartphones can be used to detect variations in altitude and to measure the absolute value of the altitude at which the device is located (altimeter function). The use of a smartphone as an altimeter requires, however, knowledge of the reference pressure and temperature to the mean sea level at the location where the phone is.

In this work we implemented a Altimeter application for Android devices equipped with barometric sensor. To obtain the pressure and reference temperature we resort to two methods: (1) the use of observation data from nearby weather stations; (2) use the prediction data of the evolution of pressure and temperature.

Finally we made a comparison of the altitude values obtained by the methods described above, comparing them also with the values obtained by the smartphone GPS receiver and the values reported by Google's location service (based on network connectivity information), present in Android smartphones.

Resumo

A inclusão de diversos sensores em *smartphones* conjugada com a capacidade de acesso à Internet veio possibilitar o desenvolvimento de aplicações com capacidades crescentes de monitorização e de atuação sobre o ambiente circundante.

O sensor barométrico, presente nalguns modelos de *smartphones*, pode ser utilizado para a deteção de variações de altitude e para a medição do valor absoluto da altitude a que se encontra o dispositivo (função altímetro). A utilização de um *smartphone* como altímetro requer, no entanto, o conhecimento da pressão e da temperatura de referência ao nível médio do mar, no local onde se encontra o *smartphone*.

Neste trabalho implementámos uma aplicação Altímetro para *Android* para dispositivos equipados com sensor barométrico. Para a obtenção dos valores de pressão e de temperatura de referência recorremos a dois métodos: (1) a utilização de dados de observação de estações meteorológicas próximas; (2) a utilização de dados de previsão da evolução da pressão e temperatura.

Após a implementação, fizemos uma comparação dos valores de altitude obtidos pelos métodos acima descritos, comparando-os também, com os valores obtidos pelo recetor Global Positioning System (GPS) do *smartphone* e com os valores reportados pelo serviço de localização da Google (baseados na informação de conectividade de rede), presente nos *smartphones Android*.

Agradecimentos

Precisava de muito espaço para agradecer a todas as pessoas, que no decorrer do meu Mestrado em Engenharia de Redes e Sistemas Informáticos me auxiliaram, diretamente ou indiretamente a concluir os meus objetivos e uma nova etapa da minha formação académica. Deste modo quero deixar o meu profundo agradecimento:

- Ao Coordenador da minha dissertação, Professor Doutor Sérgio Crisóstomo agradeço pela oportunidade e privilégio que tive pela orientação deste projeto. Fico grato pelo profissionalismo, pela amizade, e pela sua disponibilidade pois muita das vezes estando atarefado, conseguia arranjar tempo para me orientar e auxiliar.
- Aos meus colegas de Curso, e amigos, Nuno Guimarães, Marcelo Santos, Luis Barroso, Fábio Carvalho, Gil Ferro, Vladir Vicente, Jorge Silva um enorme obrigado, pela a amizade, pelo apoio e pela ajuda que foi essencial na realização da tese. Todos os dias, me fizeram erguer a cabeça, a ter calma, e não perder a esperança na conclusão desta dissertação.
- A minha Família, à minha Mãe, ao meu Pai, e ao meu Irmão, o agradecimento é imensurável, visto que eles sempre acreditaram em mim, daí me darem sempre a motivação que eu precisava para concluir esta tese. Obrigado do fundo do coração pelo vosso carinho e apoio, pois foram esses elementos essenciais que me inspiravam todos os dias para concluir esta etapa da minha vida.

  A todos eles, a minha infinita gratidão ...

Dedico esta Tese, a toda minha Família e a todos meus Amigos ...

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O objetivo principal deste trabalho consiste no desenvolvimento de uma aplicação *Android*, que permita estimar a altitude, recorrendo à informação obtida pelo sensor barométrico e por outros serviços.

Nos últimos anos, a evolução tecnológica em dispositivos móveis tem levado a um crescimento exponencial das aplicações que estes suportam. Além disso, a integração de sensores nestes dispositivos (por exemplo GPS, e acelerómetro) tem levado os *developers* a focarem-se na criação de aplicações que façam uso destas tecnologias. Um sensor que foi mais recentemente inserido nestes dispositivos foi o sensor barométrico e, consequentemente, aplicações que fazem uso desse recurso foram lançadas.

Antes do surgimento do sensor barométrico nos dispositivos móveis, o Global Positioning System (GPS) era o mais utilizado. De facto, esta tecnologia tem a capacidade de adquirir a altitude onde o dispositivo se encontra. No entanto, a precisão obtida não tem a acuidade desejada [8]. Este sensor está integrado unicamente nos dispositivos móveis mais recentes como por exemplo o Samsung Galaxy, desde a versão S3 até o S7.

Portanto, o sensor barométrico por si só não é suficiente. É necessário utilizar métodos adicionais para determinar a altitude com uma melhor precisão. Por esse motivo vamos utilizar valores provenientes de **previsões meteorológicas** armazenados em ficheiros General Regularly-distributed Information in Binary form (GRIB), como também informações de **observações meteorológicas** da superfície derivados de METeorological Aerodrome Report (METARs). Com o auxilio destes métodos, vamos obter dados como a temperatura e a pressão atmosférica, uma vez que no cálculo da altitude são elementos de grande importância.

1.1 Enquadramento e Motivação

O altímetro tem sido muito requisitado em transportes aéreos e desportos como o asa Delta, paraquedismo, montanhismo, entre outros [23]. No que consta a aviação a presença deste

equipamento é essencial porque nestas áreas onde o voo é utilizado, é sempre necessário saber uma altitude de segurança em relação ao solo e a altitude apropriada de modo a evitar trajetórias de voo de outros aviões [22].

De facto, não existem aplicações de altimetria que utilizem dados de referência provenientes de previsões e observações meteorológicas ao mesmo tempo. Deste modo, surgiu a necessidade de desenvolver uma aplicação combinando não só a informação proveniente do sensor mas também dados adicionais destas estações. Esta vai fazer o uso de ficheiros GRIB e de um *webService* para obter informações de METARs. A motivação deste projeto prende-se com a implementação de uma solução com estas caraterísticas, de modo a ter uma precisão mais confiável e estável em dispositivos móveis.

1.2 Objetivos do Trabalho

O principal objetivo deste trabalho consiste na implementação de um *software* para diapositivos móveis que permita estimar, com precisão, a altitude do utilizador. Para isso é necessário o desenvolvimento de técnicas para o aperfeiçoamento do cálculo de altitude recorrendo ao sensor barométrico apoiado por previsões (GRIB) e observações (METARs) meteorológicas.

Os seguintes objetivos secundários da dissertação são:

- Estudar técnicas de determinação de altitude através de informação de pressão atmosférica.
- Um estudo da determinação de localização através do sistema GPS.
- Investigação e compreensão do funcionamento dos sensores de GPS e barométrico.
- Implementação de uma aplicação para *smartphones Android*.
- Avaliação de desempenho.

1.3 Estrutura do Documento

Esta dissertação encontra-se estruturada da seguinte forma: No capítulo 2 é apresentado o estado de arte, onde são referidos os princípios da altimetria e métodos para determinar a altitude. O capítulo 3 é mencionado e explicado donde provêm os dados de referência que são a pressão e a temperatura. No capítulo 4 é descrito todos os passos realizados durante a implementação deste projeto, tais como o cálculo de altitude, serviço METAR e arquitetura da aplicação e do servidor GRIB. Os testes realizados com a aplicação são apresentados no capítulo 5, e por fim no capitulo 6 apresentamos as conclusões desta dissertação, referindo aspetos a melhorar e trabalho futuro.

Capítulo 2

Estado da Arte

Com o objetivo final de implementar uma aplicação para *smartphones*, foram encontrados projetos com o mesmo objetivo, mas não com os mesmos métodos de funcionamento e precisão.

De facto, atualmente já existem diversas aplicações e equipamentos com a capacidade de medir a altitude. No entanto, para uma melhor compreensão do trabalho implementado nesta tese, começamos por definir alguns conceitos essenciais.

2.1 Princípios de Altimetria

A altimetria é a ciência da medição de altitudes, bem como a interpretação dos resultados obtidos por ela. Para entender os princípios de altimetria é necessário compreender o conceito de atmosfera. A atmosfera é uma camada gasosa constituída por vários gases. Tem uma composição de cerca de 21% de oxigénio, 78% de nitrogénio e 1% de outros gases. Esta, tem o seu próprio movimento na Terra consequente das diferenças de temperaturas entre os polos e trópicos assim como das taxas de absorção e reflexão solar entre as superfícies da terra e água. Junto ao solo, a atmosfera é relativamente mais quente, dado que existe uma absorção de radiação solar por parte da água e da terra, mas à medida que subimos, a temperatura e a pressão atmosférica diminui. Portanto, a pressão é inversamente proporcional à altitude, isto é, quanto mais alto se estiver, menos atmosfera se irá ter [33].

A altitude ou altura é normalmente medida por um altímetro, geralmente em forma de um barómetro aneróide. Este tem como função registar alterações da pressão atmosférica (e consequentemente da altitude) dado que ela varia constantemente de local para local, daí este instrumento ser regulável [25].

2.2 GNSS/GPS

O GPS é um dispositivo que comunica com 4 ou mais satélites para determinar a latitude, longitude e altitude dos recetores usados pelos utilizadores determinando a posição dos mesmos.

Obter a altitude de um GPS é uma função muito complexa, uma vez que os sinais provenientes dos satélites encontram-se muitas vezes bloqueados pela Terra. Neste contexto, os recetores usam modelos matemáticos diferentes para poderem obter a altitude. Um recetor pode obter a distância a partir do centro do planeta e depois utilizar o raio da superfície para obter a altitude [8]. O GPS também conhecido como *NAVigation System with Time And Ranging Global Positioning System (NAVSTAR)* é um sistema de radio navegação baseado em satélites, que permite a qualquer utilizador saber a sua localização, velocidade, e tempo 24 horas por dia, e a sua altitude [3]. Este dispositivo é na realidade só um recetor ou componente de uma tecnologia chamada ***Global Navigation Satellite System (GNSS)***. O GNSS é um sistema de posicionamento por satélite, que se refere a uma constelação de satélites de onde são transmitidos sinais vindos do espaço, que vão ser transformados no recetor do utilizador com dados estimando a posição, informação horária e velocidade a qualquer momento e lugar na Terra.

O sistema GPS está dividido em três segmentos: o segmento espacial, o segmento de controlo e segmento do utilizador, [9] como podemos ver na figura 2.1.

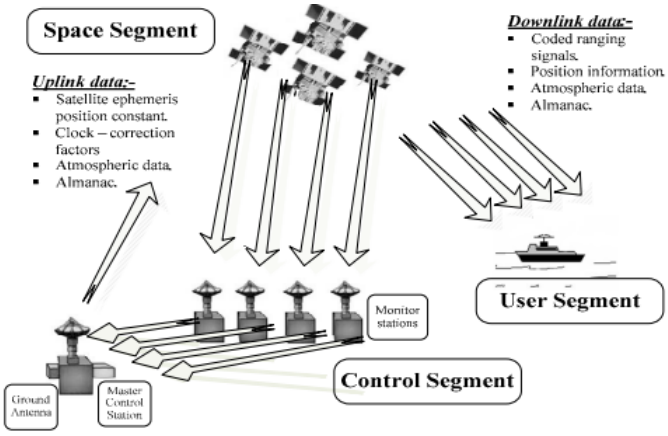

Figura 2.1: Segmentos GPS [52]

- **segmento espacial** - Consiste nos satélites GPS, que enviam sinais através do espaço.

  Este sistema é formado um grande número de satélites como está representado na figura 2.2. Estes estão distribuídos uniformemente por seis órbitas aproximadamente circulares, dando duas voltas à Terra por dia e obtendo energia a partir de painéis solares que são orientados conforme o Sol [14].

- **Segmento de controlo** - Consiste num sistema de estações de monitorização localizadas por todo o mundo. Este segmento é constituído por estações terrestres que enviam

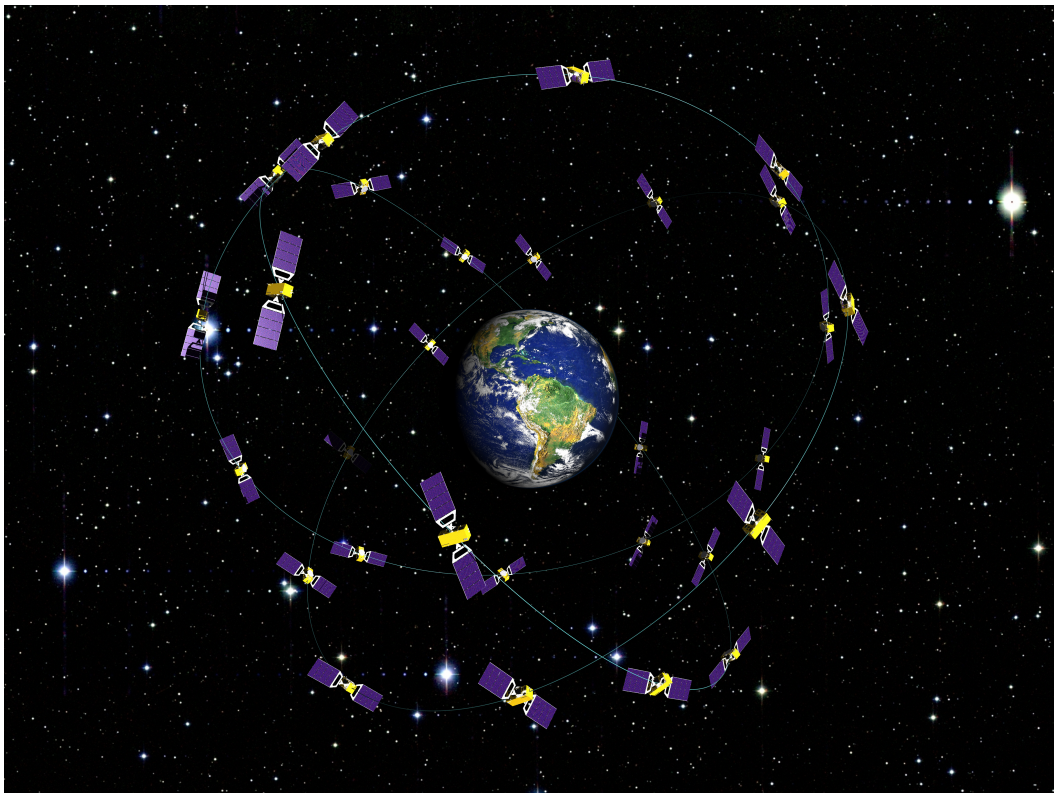

Figura 2.2: Segmento Espacial [16]

informação aos satélites, para controlarem a sua posição e velocidade, de forma a acertarem os relógios para que o tempo marcado em ambos seja equitativo. Ver figura 2.3.

Figura 2.3: Segmento de Controlo

- **Segmento utilizador** - Consiste nos recetores GPS e na comunidade dos utilizadores. Os recetores são uma unidade de processamento capaz de descodificar, em tempo real a informação enviada por cada satélite e calcular a posição do utilizador. Cada satélite envia sinais diferentes em intervalos de tempo distintos. É possível também a permuta de dados

com outros recetores ou computadores e o acesso a mapas detalhados, que permitem obter a melhor rota e visualizar as coordenadas de qualquer ponto. Ver figura 2.4.

Figura 2.4: Segmento do utilizador

Em termos de navegação, o GNSS pode ser utilizado em vários casos [34] :

- **Automóveis**: os sistemas de navegação são embutidos, onde são exibidos mapas e informações sobre a localização, velocidade, direção e pontos de interesse.
- **Navegação aérea**: têm geralmente uma exibição de mapa em movimento e são muitas vezes ligados ao piloto automático. Pode incluir também recetores GNSS montados no *cockpit* em aeronaves da aviação geral.
- **Barcos** e **navios**: usam o GNSS na navegação para definir rotas e percursos quando em alto mar.
- **Ciclismo**: este tipo de tecnologia pode ser usada em competição ou lazer, para planear ou traçar percursos com antecedência.
- **Alpinismo ou caminhadas em áreas isoladas**: usado muitas vezes para fornecer uma posição precisa do local onde estão.

2.2.1 Funcionamento do sistema de GPS

Os satélites e os recetores GPS, possuem um relógio interno com uma precisão de nano segundos, e estão constantemente a enviar sinais de rádio para o recetor [38]. Quando um satélite emite um sinal para o recetor, a hora em que ele saiu do satélite também é enviado. Resta ao recetor calcular quantos nano segundos este sinal demorou a chegar, para descobrir a posição do utilizador. Como o sinal é enviado constantemente, o GPS sabe sempre onde está o satélite mantendo assim, a posição deste sempre atualizada.

Figura 2.5: GPS [29]

O recetor recebe sinais à velocidade da luz, calculando o tempo que estes demoram a chegar até ele baseado no momento em que o satélite manda a mensagem até à chegada da mesma ao recetor. Após ter informações sobre o quão longe, pelo menos, três satélites estão, o recetor GPS pode identificar a localização do recetor utilizando um processo chamado triangulação. Como o sinal é enviado constantemente, o recetor sabe sempre onde está o satélite mantendo assim sua posição exata e sempre atualizada.

O funcionamento do GPS passa por vários pontos que são:

- Triangulação
- Distância
- Relógio
- Posicionamento
- Coordenadas
- Erros

2.2.1.1 Triangulação

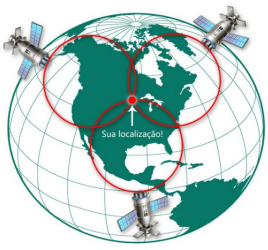

Figura 2.6: Triangulação [45]

Na triangulação um recetor GPS necessita um conjunto de 4 satélites visíveis ou mais para determinar a sua própria localização. São necessários 3 satélites para uma localização aproximada,

mas para melhorar a precisão dessa localização como por exemplo em altitude são necessários 4 ou mais. 2.6.

- O 1° **satélite** → calcula a distância de uma posição colocada numa esfera, indicando o raio.
- O 2° **satélite** → permite reduzir a incerteza a um círculo originando uma intersecção de duas esferas.
- O 3° **satélite** → gera um último círculo, que é intersectado com os dois anteriores, para mostrar o ponto mais provável da localização do utilizador.
- O 4° **satélite** → é finalmente utilizado como auxiliar. Este envia ao recetor um quarto sinal que serve de auxilio na determinação do tempo preciso. Este sinal tem como objetivo sincronizar os relógios atómicos extremamente precisos dos satélites. Deste modo um sistema de posicionamento global pode averiguar, de forma segura, a posição absoluta do utilizador. Pode também servir como substituto, caso haja uma falha num dos outros.

2.2.1.2 Distância

Para calcular a distância entre um satélite e um recetor GPS, tem que se ter em conta os seguintes parâmetros: o satélite emite um sinal que contém informação sobre a sua posição na órbita e a "hora **t**", marcada no seu relógio atómico. O recetor recebe este sinal no instante $\mathbf{t + \Delta t}$ no seu relógio. Como o sinal se desloca à velocidade da luz ($\mathbf{c = 299\ 792\ 458\ m\ /\ s}$), o recetor calcula a distância, **d**, que o separa do satélite, através da expressão [30]:

$$\mathbf{d = \Delta t \times c}$$

2.2.1.3 Relógio

O GPS requer um sistema de sincronização de satélites. Nesta são necessários relógios extremamente precisos, pois os satélites têm que estar sincronizados perfeitamente para o envio de dados entre eles.

No que consta à localização, os satélites são lançados em órbitas precisas. No entanto os recetores usam um *almanac*. O *almanac* é um conjunto de dados que cada satélite transmite, e que inclui informações sobre o estado dos próprios, não só de um, mas sim de toda a constelação de satélites enviados.

2.2.1.4 Posicionamento

A posição de um recetor na Terra é referenciada em relação ao equador e ao meridiano de Greenwich e traduz-se por três números: A latitude, a longitude e a altitude. Para saber a

posição de um recetor GPS utilizado por um utilizador na Terra são necessários saber estes dados. As latitudes e longitudes são coordenadas geográficas. A latitude mede-se para norte e para sul do equador, entre 90º sul, no Pólo Sul e 90º norte, no Pólo Norte como se pode observar na figura 2.7.

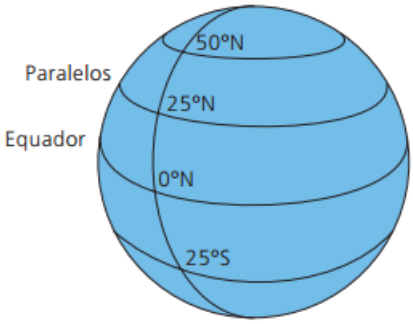

Figura 2.7: Latitudes [49]

A longitude é medida ao longo do Equador, e representa a distância entre um ponto e o Meridiano de Greenwich. Também é medida em graus, podendo ir de 0º a 180º para Leste ou para Oeste, como está ilustrado na figura 2.8.

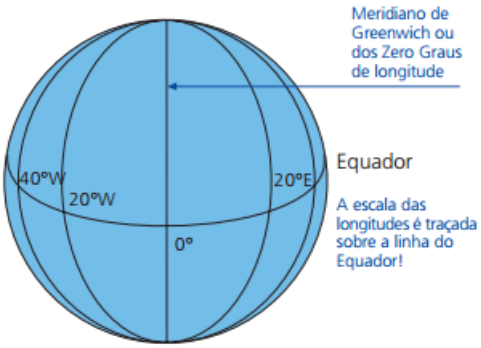

Figura 2.8: Longitudes [50]

A Terra é aproximadamente esférica, com um ligeiro achatamento nos polos. Para se definir a altitude de um ponto sobre a Terra define-se uma esfera geóide com um raio de 6378 km. A altitude num ponto da Terra é a distância na vertical à superfície deste geóide [18] .

2.2.1.5 Erros

O GPS tem as suas vantagens mas também têm alguns erros que necessitam ser corrigidos.

- **Erros de órbita**: São imprecisões na localização divulgada pelo satélite.

- **Atrasos na Troposfera e Ionosfera**: Atrasos do sinal ao atravessar as camadas da atmosfera terrestre.
- **Multi percursos**: Erros devido à receção de mais que um sinal proveniente da mesma fonte.

2.3 Determinação da altitude através da Pressão Atmosférica

A pressão atmosférica é a força que o ar exerce sobre as superfícies com as quais tem contacto. Esta força é normalmente medida em milibares (mb) com a ajuda de um barómetro, sendo o valor médio da pressão da Terra de 1013 milibares ou 760 milímetros de mercúrio (mmHg) [44]. A pressão varia consoante a elevação vertical e a temperatura de uma localização, pois quanto maior for a altitude de um relevo em relação ao nível do mar, menor será a pressão atmosférica. Da mesma forma, quanto mais baixa for a altitude, maior será a pressão. Este processo ocorre devido à força da gravidade que sustem a maior parte do ar mais próximo da superfície [2]. A tabela 2.1 apresenta variações da pressão atmosférica em relação à altitude.

VARIAÇÃO DA PATM COM A ALTITUDE

Altitude (m)	Pressão atmosférica (mmHg)	Altitude (m)	Pressão atmosférica (mmHg)
0	760	1200	658
200	742	1400	642
400	724	1600	627
600	707	1800	612
800	**690**	2000	598
1000	674	3000	527

Tabela 2.1: Exemplo de valores da pressão que alteram com a altitude

A temperatura é um fator importante na pressão, visto que consegue altera-la em relação aos seus níveis de intensidade. Por exemplo, quando o ar aquece, as moléculas agitam-se com mais intensidade e com isso o ar tende a expandir, ficando mais leve e criando zonas de baixa pressão atmosférica.

Quando se dá o efeito inverso, existe uma menor agitação de moléculas causando a contração do ar e consequentemente a aproximação destas. Por esse motivo, o ar acaba por ficar mais concentrado e pesado, criando zonas de alta pressão atmosférica. Esta situação pode ser verificada na figura 2.9.

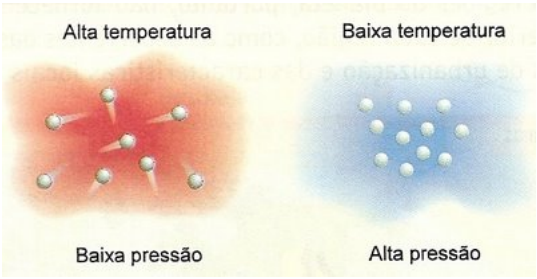

Figura 2.9: Pressão e Temperatura [37]

2.3.1 Altimetria baseada em pressão atmosférica

A altimetria ou nivelamento, tem como função determinar a distância vertical ou diferença de nível entre diversos pontos [32]. Existem muitos tipos de nivelamento, tais como o geométrico, o trigonométrico, o taqueométrico, e o barométrico. O nivelamento utilizado neste trabalho é o barométrico. Este, permite determinar a altitude de um ponto através da pressão atmosférica baseada na relação inversamente proporcional entre pressão e altitude [15].

Como foi referido o ar tem um peso, e é esse que é denominado como pressão. Para medir esta força ou peso, é utilizado um barómetro como o que está representado na figura 2.10. Existem vários estudos que comprovam que esta ferramenta fornece uma medida muito mais precisa da altitude do que o recetor GPS do *smartphone* [4].

A pressão normalmente pode ser medida em Hpa (Hectopascais) ou em mb (milibares) e no caso de aviação é lida em pés, no qual um pé corresponde a 0.3048 metros.

Figura 2.10: Barómetro [2]

A altitude é calculada pela seguinte fórmula matemática usada pela International Civil Aviation Organization (ICAO) e pela International Standard Atmosphere (ISA) [19]:

$$h = \frac{T_0}{L}\left(1 - \left(\frac{p}{p_0}\right)^{\left(\frac{RL}{gM}\right)}\right)$$

onde:

- h – altitude acima do nível médio do mar [m]
- t_0 – temperatura base ao nivel médio do mar[°K]
- L – gradiente adiabático: 0.0065 °K/m
- p – pressão atmosférica ambiental [hPa]
- p_0 – pressão base ao nível médio do mar [hPa]
- R – constante universal dos gases perfeitos: 8.31432 N·m/(mol·K)
- g – gravidade da Terra: 9.80665 m/s2
- M – massa molar do ar: 0.0289644 kg/mol

Um avião utiliza um altímetro do género da figura 2.11, que tem geralmente a forma de um barómetro aneróide. O principal objetivo deste dispositivo é registar alterações de pressão atmosférica que acompanham as variações de altitude.

Como a pressão está constantemente a variar de local para local, este instrumento é regulável. Deste modo é possível ajustar a pressão de maneira a poder ler corretamente a altitude, através da janela de Kollsman.

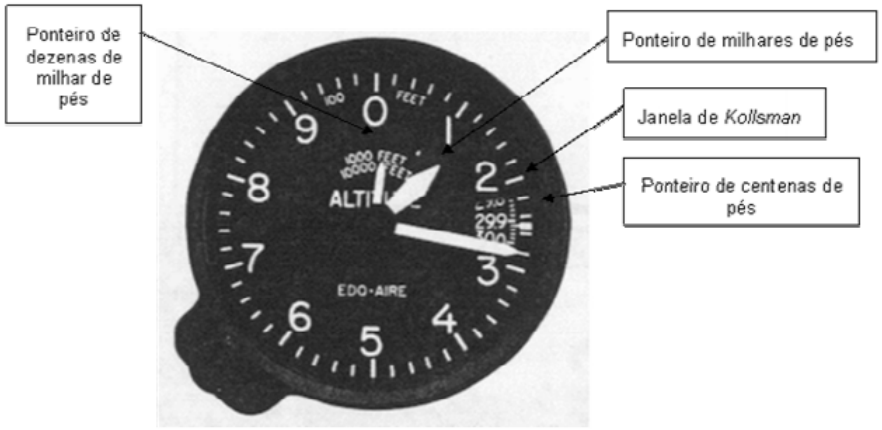

Figura 2.11: Altímetro [28]

Como já foi referido, no cálculo da altitude é necessário a pressão atmosférica. Esta pode ser medida com um barómetro. No entanto, alguns problemas na utilização destes aparelhos [36]:

- **Densidade de pontos de referência**: A precisão da altura num ponto depende da distância entre um ponto e o ponto de referência. Os únicos pontos de referência disponíveis são estações meteorológicas que são dificilmente localizadas.

- **Calibração**: Este equipamento necessita calibração.
- **Precisão**: Existem barómetros com menor precisão e outros com maior.

Os métodos para calcular a altitude usam valores de referência que neste caso vão ser a pressão e a temperatura.

Capítulo 3

Pressão e Temperatura de referência

Nos métodos utilizados, são necessários dados de referência. Estes referem-se à pressão e à temperatura e são aplicados no cálculo da altitude. Os valores destas duas variáveis foram retirados de ficheiros GRIB pertencentes a um site com previsões meteorológicas, e de um *Web Service* de METeorological Aerodrome Report (METARs) com informações meteorológicas atualizadas de um sistema aéreo mundial.

3.1 METARS (METeorological Aerodrome Report)

Os METARs são relatórios de observações meteorológicas dedicados à aviação, que contêm uma grande quantidade de informação sobre observações codificadas de um aeródromos especificos [48]. Um exemplo de um desses relatórios pode ser visto na tabela 3.1.

3.1.1 Como interpretar os códigos de um METAR

Normalmente um relatório meteorológico é repartido em 9 partes:

- **Tipo de Reporte** - Origem da observação, exemplo (METAR ou SPECI).
- **Estação/Data e Hora** - A estação aparece em código *ICAO* que pertence a uma observação em forma de quatro letras, juntamente com a representação do dia e da

Tipo de Reporte	Estação / Data e Hora	Vento	Visibilidade	Tempo Presente	Condições do céu	Temperatura / Ponto de Orvalho	Pressão (QNH)	Observações (Remarks)
METAR	LPLA 131900Z	20022G36KT	4000	-RA	SCT012 BKN018 OVC080	21/17	Q1007	GRN
METAR	LPMA 131930Z	02004KT 350V060	9999		SCT018	19/13	Q1022	RS3405KT 053604KT 230103KT

Tabela 3.1: Reporte Metereológico com 2 METARs

hora em *Universal Time Coordinated (UTC)*.

As quatro letras estão divididas da seguinte forma: A 1° letra representa uma área no mundo. Na figura 3.1 podemos ver como se designam todas essas áreas.

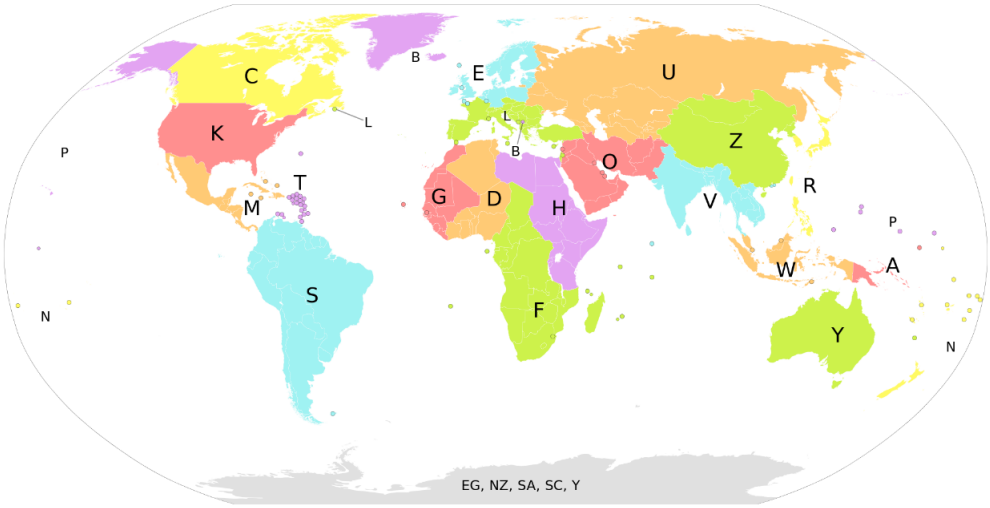

Figura 3.1: Areas representadas pelos METARs [54]

A 2° letra determina o país enquanto as 3° e 4° letras são os nomes dos aeródromos ou regiões onde eles estão localizados.

Nos números que se seguem, os primeiros dois dígitos representam o dia, e os quatro logo a seguir, a hora e os minutos. A última letra representa o fuso horário. Neste caso *Z* refere-se ao UTC.

- **Vento** - representa a direção e a velocidade do vento, em que os primeiros três dígitos representam a direção do vento em graus, e os últimos dois, a velocidade do vento representada em nós (kt). Umas representações podem vir com a letra **V**, isto indica que a direção varia.

- **Visibilidade** - A visibilidade está relacionada com a opacidade do campo de visão, vindo indicada em metros ou em milhas. Os códigos de visibilidade variam entre 0000 (quando é inferior a 50 m) e 9999 (quando é superior ou igual 10 km).

- **Tempo Presente** - é normalmente representado por um conjunto de duas letras com um significado específico. A descrição para cada conjunto pode ser consultado na tabela 3.2.

- **Condições do céu** - Representa através de um acrónimo e uma medição com variação em oitavos (entre 0/8 e 8/8), a presença de nuvens no céu onde:

  - 0/8 Céu Limpo (*Sky Clear* – SKC)

  - 1/8 a 2/8 Pouco Nublado (*Few* – FEW)

  - 3/8 a 4/8 Pouco Nublado (*Scattered* – SCT)

  - 5/8 a 7/8 Muito Nublado (*Broken* – BKN)

  - 8/8 Encoberto (*Overcast* – OVC)

INTENSIDADE OU PROXIMIDADE 1	CARACTERÍSTICAS (DESCRITOR) DO FENÓMENO 2	PRECIPITAÇÃO 3	OBSCURECIMENTO 4	OUTROS 5
(- Fraco)	MI Pouco espesso	DZ Chuvisco	BR Neblina	PO Turbilhão bem desenvolvido de poeira ou areia
	BC Bancos	RA Chuva	FG Nevoeiro (- de 1000m)	
Moderado (sem qualificador)	PR Parcial (Cobrindo parte do aérodromo)	SN Neve	FU Fumo	SQ Borrasca
(+ Forte)	DR "abaixo de 2m"	SG Neve em grãos	VA Cinzas vulcânicas	FC Tornado ou tromba de água
	BL "acima de 2m"	IC Prismas de gelo	DU Poeira	SS Tempestade de areia
VC (na vizinhança)	SH Aguaceiro(s)	PE Granizo	SA Areia	
	TS Trovoada	GR Saraiva	HZ Bruma	DS tempestade de poeira
	FZ Gelado	GS Granizo (Translúcidas)		

Tabela 3.2: Tabela com a significação de cada letra do tempo presente

- **Temperatura** - Variação da temperatura.
- **Pressão** - Pressão atmosférica medida pela estação (aeródromo).
- **Observações** - Informações adicionais como altura e visibilidade da superfície.

3.2 GRIB (General Regularly-distributed Information in Binary form)

O GRIB é um formato de dados binários usados em meteorologia para armazenar informações referente a previsões meteorológicas. Este tipo de ficheiro é usado pelos institutos para visualizar, manipular e armazenar essas informações [21]. Existem duas versões o GRIB1 e o GRIB2, cada um deles com princípios de codificação semelhantes, mas com uma estrutura diferente.

Em comparação com o GRIB1, a nova versão é mais flexível (por causa da sua estrutura), possui variáveis com mais precisão, bem como uma descrição de dados mais complexos [41].

No que consta a estrutura, um destes ficheiros pode conter uma ou mais mensagens GRIB e cada uma destas mensagens contém várias secções como podemos ver no esquema representado na figura 3.2.

As Aplication Programming Interface (API)s existentes e disponíveis para a descodificação destes ficheiros podem variar entre **C**, **Fortran 90** e **Python**. Todas estas fornecem uma maneira de fazer um `get/set` da mesma chave e valores de mensagens estruturadas dentro de cada um dos ficheiros.

A principal função de uma GRIB *API* é descodificar e codificar ambas as versões com as mesmas funções, bem como fornecer um modo de converter os dados entre as duas. A API necessita de ser flexível o suficiente para fazer facilmente os *updates* para novas tabelas e *templates*,

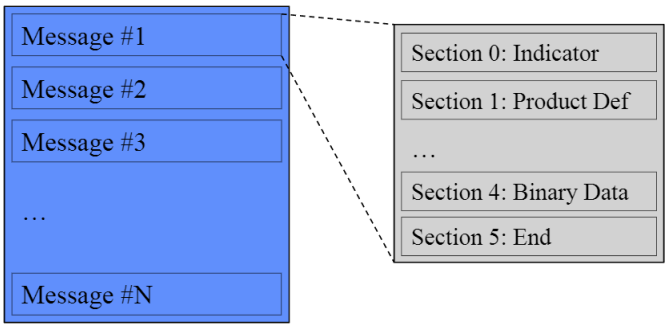

Figura 3.2: Estrutura do GRIB [40]

da versão antiga dos ficheiros para a mais recente. É possível obter estes ficheiros de duas maneiras: fazer o *download* do arquivo via Internet e solicitar e receber o ficheiro GRIB através de *e-mail*. Normalmente identifica-se os ficheiros GRIB através da sua extensão, podem ser .grb ou .grib2.

Para a extração e interpretação dos ficheiros GRIB foi utilizada a linguagem Python complementado com o *package* pyGrib.

3.3 Aplicações

A aplicação a ser desenvolvida vai-se diferenciar das mencionadas a seguir visto que vai ser implementada uma abordagem diferente à utilizada. Enquanto a maioria das aplicações já existentes usam apenas dados provenientes de uma fonte (*GPS* ou sensor barométrico), a aplicação a ser desenvolvida vai combinar a informação proveniente desses mesmos sensores com a obtida de ficheiros *GRIB* e *METARs*.

Esta abordagem está a ser baseada na premissa das medições de altitude barométrica, que fornecem informações mais precisas sobre mudanças de altitude relativamente ao GPS. No entanto, uma estimativa precisa da altitude absoluta com base em medições de pressão barométrica requer calibração dependendo dos locais situados e devido a mudanças naturais na pressão atmosférica causada pelas condições meteorológicas.

3.3.1 Visualizadores

Existem muitos visualizadores de ficheiros GRIB e estes são necessários para a conversão das informações para um formato gráfico convencional com zonas costeiras de latitude e longitude. Um dos visualizadores utilizados e estudados foi o **Panoply**. Este visualizador é na realidade uma aplicação multi-plataforma que traça matrizes geo-reticuladas de Network Common Data Form (netCDF), Hierarchical Data Format (HDF) e conjuntos de dados GRIB [47]. Também permite obter uma boa compreensão dos ficheiros ao permitir seccionamento de áreas em especifico.

Este programa foi uma grande ajuda para analisar ficheiros deste tipo, visto que ele permite a visualização de dados da pressão e de temperatura relativamente à latitude, longitude, hora e data introduzida. Isto permitiu o *debug* dos *scripts* implementados em Python (e mencionados no capítulo seguinte) ao comparar os resultados obtidos destes com os visualizados no Panoply.

Outros visualizadores normalmente usados para analisar este tipo de ficheiros são: Saildocs Viewfax, ExpeditionLT, GribView, Ugrib, Zygrib e OCENS GRIB Explorer.

Através de um pequeno estudo, estes visualizadores apresentam ter na sua maioria **isobars**, **windbars**, **círculos de precipitação**, e divisões com cores variadas que indicam variações de **temperatura**.

3.3.2 PocketGrib

Uma aplicação útil para o desenvolvimento deste trabalho foi o **PocketGrib**, com esta aplicação *Android* é possível visualizar e fazer *download* de arquivos GRIB, analisando dados climáticos globais. O *download* é direto, possuindo dados GRIB do modelo *Global Forecast System (GFS)* da *National Oceanic and Atmospheric Administration (NOAA)*, com a possibilidade de realizar previsões até 8 dias, estes dados são exibidos numa simples interface [**?**].

Figura 3.3: PocketGrib Map [7]

Esta é uma aplicação muito útil não só para marinheiros, como também para *windsurfers*, caçadores de tempestades, meteorologistas amadores e profissionais. Estes ficheiros do Sistema Global da NOAA contêm dados meteorológicos como: Velocidade e direção do Vento, precipitação, pressão, temperatura e altura das ondas.

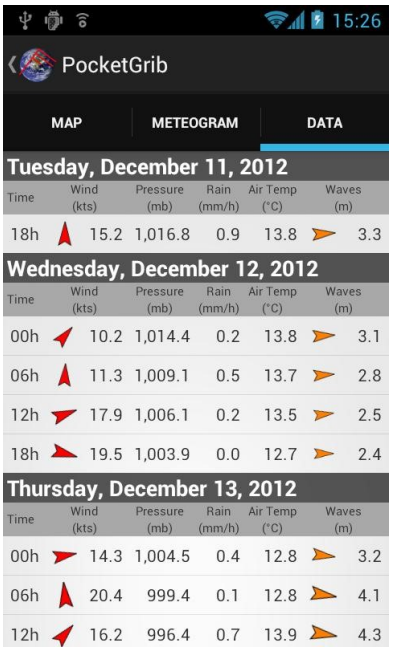

Figura 3.4: PocketGrib Results [6]

3.3.3 Altímetro preciso livre

É outra aplicação desenvolvida para dispositivos móveis. Utiliza três métodos no seu funcionamento para estimar a altitude:

- A triangulação dos satélites GPS.
- A elevação do solo na sua posição atual.
- Se disponível usa o sensor de pressão do *smartphone*.

Esta aplicação tem algumas vantagens como o funcionamento sem ligação à Internet e caso esta exista, calibra-se de acordo com a pressão da estação meteorológica mais próxima para melhorar a precisão. No entanto, também apresenta alguns pontos fracos como o facto do método do GPS ser lento e não muito preciso.

3.3.4 Runtastic Altimeter

Esta aplicação de altimetria, é utilizada maioritariamente por amantes dos desportos e atividades ao ar livre. Este calcula a altitude através do GPS.

Esta aplicação tem a vantagem de ter muitos recursos como:

- Medição precisa da altitude (elevação) com o serviço online Runtastic que minimiza a hipótese de erros.

- Função de foto com altitude integrada e partilha.
- Compartilhar fotos com amigos nas redes sociais e serviços de mensagens como o Google+, Facebook, Twitter, WhatsApp ou *e-mail*.
- Indicação de altitude em metros ou pés.
- Informações sobre os horários do nascer e do pôr do sol.
- Bússola.
- Coordenadas através do GPS.
- Temperatura e velocidade do vento atual.

As principais desvantagens são:

- O desaparecimento de coordenadas no sistema de fotografias.
- Erros nas coordenadas e valores exagerados de temperatura e velocidade do vento.

Depois do estudo de algumas aplicações e nomeação de alguns visualizadores existentes, vai ser demonstrado no capítulo "Especificação" a arquitetura da solução proposta, a sua implementação, e como esta comunica com o servidor.

Capítulo 4

Especificação e implementação

Neste capítulo vão ser mencionadas as soluções usadas para o cálculo da altitude que são: os dados retirados das previsões pertencentes aos ficheiros *GRIB* e as observações vinculadas aos *METARs*. Para cada solução vão ser mencionados todos os passos realizados para obter os dados e qual o tratamento realizados nesses dados. Para além disto, vai ser descrito a arquitetura do servidor GRIB e da aplicação, explicando como é que a comunicação entre ambos é realizada.

Na arquitetura do servidor vamos detalhar como foi feito o processamento da informação, a sua interpretação e o seu processamento. Na aplicação vai ser detalhado cada serviço utilizado pela mesma, por exemplo, *GPS*, sensor barométrico, METAR e o serviço da Google.

4.1 Soluções para o cálculo da Altitude

Uma das soluções para o cálculo da altitude, consiste em analisar os dados estruturados dentro de cada ficheiro GRIB. Os dados necessários são os seguintes:

- Pressão (*Mean sea level pressure*).
- Temperatura (*2 metre temperature*).

Para analisar estes dados, usamos a linguagem de programação Python, sendo a mesma uma das opções mencionadas pela European Centre for Medium-Range Weather Forecasts (ECMWF) para fazer o *GET* e o *SET* das mensagens, chaves e dos valores dos ficheiros [53].

Como ajuda para extração dos dados dos ficheiros GRIB, usamos as seguintes bibliotecas: `pygrib` e `numpy` [17]. O `numpy` é uma bilioteca requisitada pelo `pygrib`, e tem a capacidade de suportar *arrays* e matrizes multi-dimensionais para a possível extração dos dados necessitados.

No caso do METAR, o procedimento foi relativamente parecido aos ficheiros GRIB, em vez da utilização de ficheiros transferidos por estações meteorológicas, foi usado `Web Service` fornecido

pela *Aviation Weather Center* que possui um ficheiro eXtensible Markup Language (XML) com toda a informação necessária.

4.2 Arquitetura

Nesta secção serão descritas as componentes de *software*, propriedades externas, relacionamento com os equipamentos e os *softwares* utilizados.

4.2.1 Visão Geral

Nesta subsecção vai ser feita uma visão geral do todo o percurso para a elaboração da aplicação. Inicialmente foi necessário extrair os dados dos ficheiros GRIB, através da realização de *scripts* Python.

Foi necessário fazer a interpolação temporal e espacial, uma vez que foi necessário conjugar latitudes, longitudes e o tempo, com as pressões e as temperaturas. A interpolação espacial e temporal são processos que utilizam pontos com valores conhecidos para estimar pontos desconhecidos.

Após os *scripts* que tratam dos dados obtidos por serviços externos estarem concluídos, foi necessário preparar os servidor para receber pedidos por parte da aplicação do dispositivo móvel. Essa comunicação é feita através de `sockets`. A função deste servidor é esperar que a aplicação faça um pedido para retornar os dados obtidos dos serviços externos.

Como *framework* para o desenvolvimento da aplicação do smartphone foi utilizado o **PhoneGap**, que permite desenvolver aplicação móveis de maneira híbrida, pois utiliza combinações de tecnologias `Web`, como *HyperText Markup Language (HTML)*, *Cascading Style Sheets (CSS)* e JavaScrip [26].

Foi usada outra *framework*, chamada **Ionic**, baseada na criação de aplicações híbridas para os dispositivos móveis, constituída por vários componentes[46]. Estes componentes são:

- Cordova: Integração com recursos nativos dos dispositivos.
- AngularJS: Criação da parte Web da Aplicação.
- Ionic Module: Ferramentas e Componentes disponibilizados pela *framework*.

Em termos de pré-requisitos para usar o **Ionic** e desenvolver estas aplicações com o Cordova/Phonegap é necessário ter instalado o **NodeJS** e o NPM. O **NodeJS** é uma plataforma do lado do servidor construído no motor JavaScript do Google Chrome para o desenvolvimento de aplicações de rede. O NPM é um gestor de pacotes da mesma linguagem que encontra, compartilha e reutiliza pacotes de código [12]. Após estes sistemas estarem em sintonia uns com

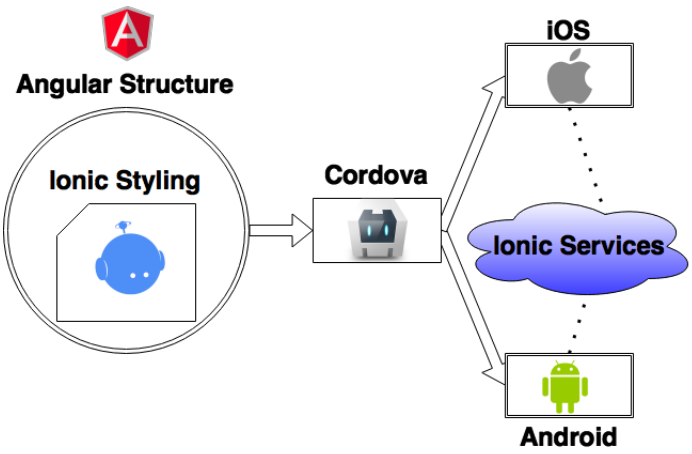

Figura 4.1: Diagrama de explicação Ionic [31]

os outros, foram instalados os *plugins* para o funcionamento do GPS, do *socket*, e do barómetro. Com ajuda destes *plugins* é possível obter dois tipos de altitude, a do GPS e a proveniente do cálculo dos dados recebidos pelo servidor.

Conseguiu-se obter a altitude do METAR através do tratamento dos dados lidos no ficheiro XML. Neste sistema foi necessário acrescentar um serviço que facilita a comunicação com os servidores disponibilizados pelo METAR. Esta comunicação é possível com um serviço do `AngularJS`, chamado de `$http`.

4.2.2 Servidor Grib

Como já referido na secção anterior, a aplicação comunica com o servidor através de *sockets*. Usamos o `WebSocket`, que é um protocolo de comunicação bidirecional entre cliente e servidor. A diferença entre um `socket` e um `WebSocket`, é que um `socket` é utilizado para comunicação entre processos através da rede, transmitindo dados em formato binário. O `webSocket` é um protocolo de comunicação, proporcionando canais de comunicação full-duplex sobre uma conexão TCP, Este opera sobre os protocolos *Transmission Control Protocol (TCP)* ou *User Datagram Protocol (UDP)* [13].

O *handshake* que inicia a comunicação é muito parecido com o do HTTP, daí os servidores `Web` podem facilmente servir de servidores HTTP e `WebSocket` na mesma porta. Este protocolo usa o método de comunicação *three-way-handshake* representado na figura 4.2.

O servidor usa como sistema operativo Linux, mais concretamente a distribuição Lubuntu. Para que a comunicação com o servidor seja fazível em qualquer lugar, configurou-se o serviço `dynip`. Este serviço é fornecido pelo Internet Service Provider (ISP), e basicamente cria um nome associado ao endereço Internet Protocol (IP) público da rede onde se encontra o servidor. No *router* da rede do servidor, redirecionou-se a porta 8000 para o endereço IP privado do servidor. Desta forma, todos os pedidos que chegam da Internet com a porta de destino 8000 são

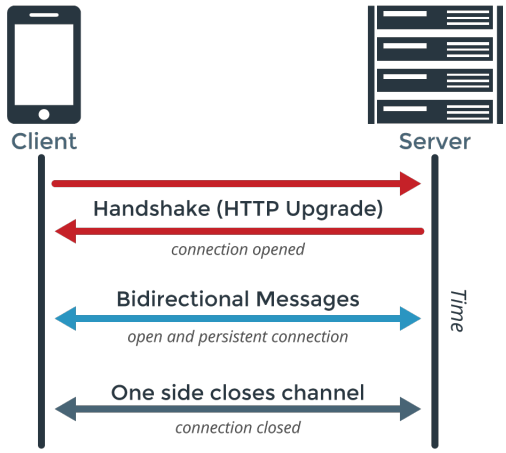

Figura 4.2: *Simple Handshake Diagram [24]*

reencaminhados para o servidor.

As mensagens que o servidor e o cliente trocam são do tipo ***JavaScript Object Notation (JSON)*** que é uma formatação leve de troca de dados. Para o utilizador torna-se mais fácil para ler e escrever, e para as máquinas é mais fácil de interpretar e gerar [5]. O servidor está à espera de um pedido por parte do cliente. Caso a ligação seja bem sucedida, o cliente envia os dados para o servidor, que responde com os dados obtidos dos ficheiros. [39]. Uma vez que a comunicação com o servidor acaba, o canal de comunicação é fechado. A figura 4.3 representa todos os passos mencionados.

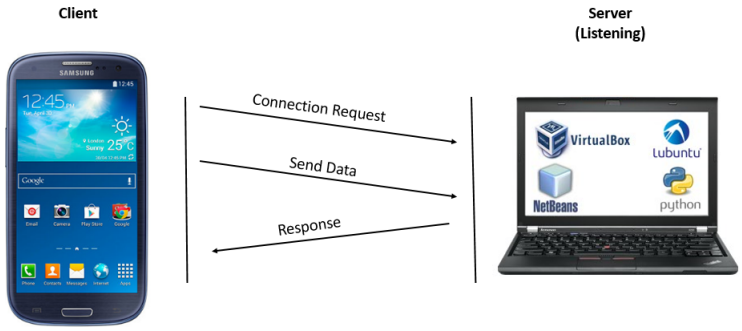

Figura 4.3: *Socket Communication*

4.2.3 Aplicação Cliente

Para o desenvolvimento da aplicação, foi usado o **PhoneGap**, uma *framework* de desenvolvimento de *software*, que é utilizada no desenvolvimento de aplicações móveis. Para desenvolver estas aplicações usando o **PhoneGap**, o programador não precisa ter conhecimentos da linguagem de programação móvel, mas sim, de linguagens de programação `Web`.

Para criar a aplicação basta seguir os passos representado na figura 4.4. No fim destes passos é possível ver a aplicação criada no *browser*, como podemos observar na figura 4.5.

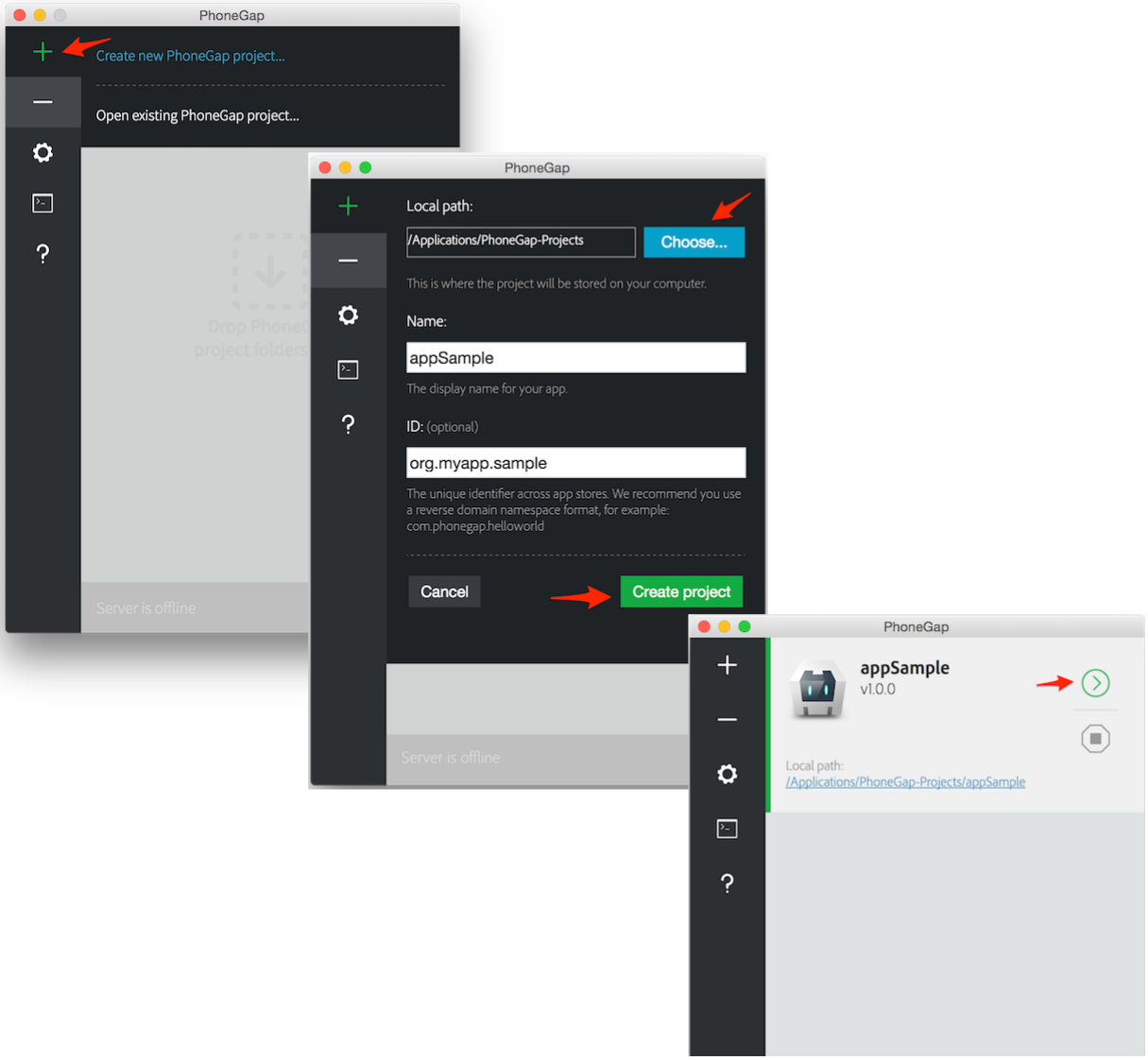

Figura 4.4: *Create app project*

As vantagens do **PhoneGap** em relação aos outros métodos são:

- Facilidade para manipular componentes nativos da plataforma.
- Multi-plataforma.
- Não é necessário aprender a linguagem de programação usada pela sistema operativo móvel.
- Compatibilidade com várias *frameworks*.

A desvantagem mais relevante do **PhoneGap**, é a possibilidade de não existir *plugins* para manipular certos componentes no dispositivo móvel.

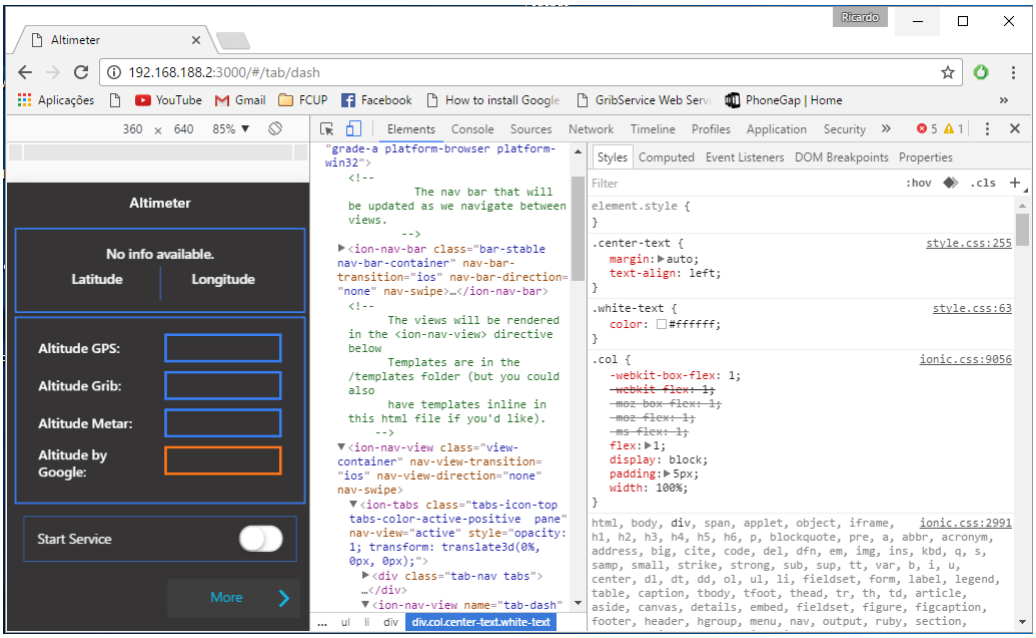

Figura 4.5: *Exemplo de uma phonegapp app criada*

Foi ainda instalado o **Ionic**, um poderoso *Software development kit (SDK)* HTML5 mais voltado para a aparência e interação *User Interface (UI)* da aplicação, que ajuda na construção de aplicações móveis usando tecnologias `Web`, como HTML, CSS, e JavaScript. O **Ionic** exige atualmente uma *framework* JavaScript *open-source*, chamada **AngularJS**. Para criar uma aplicação deste tipo é necessário instalar uma *framework* baseada em JavaScript, aconselhada pela **Ionic**, nomeada de **NodeJS**. A plataforma utilizada nesta aplicação foi o *Android* daí ser necessário instalar o SDK indicado.

4.2.4 METAR

Como já foi referido os METARs são observações com base na meteorologia que possui uma grande quantidade de informação codificada. Neste serviço a informação codificada foi retirada de um ficheiro XML proveniente de um `Web Service` da *Aviation Weather Center*.

Para obter esse XML, foi utilizado o serviço `$http` do **AngularJS** que facilita a comunicação com os servidores HTTP via o `XMLHttpRequest` do navegador [11]. O serviço `$http` é uma função que recebe um único argumento, utilizado para gerar um pedido HTTP devolvendo um valor que pode ser retornado futuramente ou não, chamado de promessa. O `XMLHttpRequest` é uma *API* que fornece funcionalidades de cliente para transferência de dados entre um cliente e um servidor [42]. Fornece uma maneira fácil de obter dados de um *Uniform Resource Locator (URL)* sem ter que fazer uma atualização de página inteira. Isso permite a uma página `Web` atualizar apenas uma parte da página sem interromper o que o utilizador está a fazer. O `$http` baseia-se na API *deferred/promise* do serviço `$q` que ajuda a executar funções de forma assíncrona, ou seja, utiliza valores de retorno (ou exceções) quando acabam o processamento. Basicamente este

serviço descreve uma promessa como uma interface para interagir com um objeto. Uma promessa representa um resultado de uma ação executada de forma assíncrona, e pode ou não ser concluída em um determinado ponto no tempo [43]. Como os dados eram recebidos em XML foi necessário adicionar uma bibloteca designada de `x2js` para poder fazer a conversão para JSON [1].

Este `Web Service` tem algumas regras:

- A longitude inserida não pode ser menor a -180 e maior que 180 graus.
- A latitude inserida não pode ser menor a -90 e maior que 90 graus.
- A distância máxima para a localização dos METARs é de 800 km.

Dentro de cada METAR é possível encontrar informações como o *Identificação Digital (ID)* de cada estação, a data da última observação, o local da observação referindo a sua latitude, longitude, pressão, temperatura, entre outros, como se pode ver na figura 4.6.


```
▼<response xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance" version="1.2"
 xsi:noNamespaceSchemaLocation="http://aviationweather.gov/adds/schema/metar1_2.xsd">
  <request_index>261980096</request_index>
  <data_source name="metars"/>
  <request type="retrieve"/>
  <errors/>
  <warnings/>
  <time_taken_ms>13</time_taken_ms>
 ▼<data num_results="10">
  ▼<METAR>
   ▼<raw_text>
      KBKF 261758Z VRB02KT 10SM SCT110 SCT220 19/08 A3017 RMK AO2A SLP161 T01890077 10189 20123 58007
    </raw_text>
    <station_id>KBKF</station_id>
    <observation_time>2016-08-26T17:58:00Z</observation_time>
    <latitude>39.72</latitude>
    <longitude>-104.75</longitude>
    <temp_c>18.9</temp_c>
    <dewpoint_c>7.7</dewpoint_c>
    <wind_dir_degrees>0</wind_dir_degrees>
    <wind_speed_kt>2</wind_speed_kt>
    <visibility_statute_mi>10.0</visibility_statute_mi>
    <altim_in_hg>30.171259</altim_in_hg>
    <sea_level_pressure_mb>1016.1</sea_level_pressure_mb>
   ▼<quality_control_flags>
      <auto_station>TRUE</auto_station>
    </quality_control_flags>
    <sky_condition sky_cover="SCT" cloud_base_ft_agl="11000"/>
    <sky_condition sky_cover="SCT" cloud_base_ft_agl="22000"/>
    <flight_category>VFR</flight_category>
    <three_hr_pressure_tendency_mb>-0.7</three_hr_pressure_tendency_mb>
    <maxT_c>18.9</maxT_c>
    <minT_c>12.3</minT_c>
    <metar_type>METAR</metar_type>
    <elevation_m>1726.0</elevation_m>
  </METAR>
 ▶<METAR>...</METAR>
 ▶<METAR>...</METAR>
```


Figura 4.6: Dados de METARs no ficheiro XML

4.3 Especificação do funcionamento do servidor GRIB e sua Implementação

4.3.1 Obtenção dos Modelos

O servidor precisa de ter os ficheiros GRIB constantemente atualizados, para isso foi desenvolvido um *script* que trata de fazer *download* do ficheiro atualizado.

Os *downloads* são feitos através de um site, que possui bastantes dados meteorológicos chamado de **Meteo France**. Todos os dias estão disponíveis ficheiros novos e atualizados de 6 em 6 horas. Estes ficheiros dividem os dias de 6 em 6 horas pela seguinte ordem, 00H, 06H, 12H, 18H e novamente 00H. São atualizados 5 horas depois da hora prevista, ou seja, supondo que a hora atual é 12H, o ficheiro só vai estar disponível por volta das 17H. Cada um destes ficheiros são transferidos através do URL referido no site, e são guardados com o nome da data e hora em que são processados.

Este *script* tem também o objetivo de limpar os ficheiros mais antigos. O *script* percorre a pasta onde estão guardados os ficheiros e elimina aqueles com mais de 48H.

4.3.2 Interpretação da informação

Os ficheiros GRIB estão organizados numa certa estrutura, para facilitar a leitura dos dados necessário, foi usada uma biblioteca para Python, designado por `pygrib`. Esta biblioteca trabalha com ficheiros GRIB da primeira e da segunda versão. Como se pode ver nas figuras 4.7 e 4.8, os dados mais importantes são o *Mean sea level pressure* e o *2 metre temperature*.

```
1:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 0 hrs:from 201606060000
2:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 1 hrs:from 201606060000
3:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 2 hrs:from 201606060000
4:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 3 hrs:from 201606060000
5:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 4 hrs:from 201606060000
6:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 5 hrs:from 201606060000
7:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 6 hrs:from 201606060000
8:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 7 hrs:from 201606060000
9:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 8 hrs:from 201606060000
10:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 9 hrs:from 201606060000
11:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 10 hrs:from 201606060000
12:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 11 hrs:from 201606060000
13:Mean sea level pressure:Pa (instant):regular_ll:meanSea:level 0:fcst time 12 hrs:from 201606060000
```

Figura 4.7: Dados GRIB estruturados (pressão)

Ressalva-se, ainda, a possibilidade de dentro desses ficheiros existirem dados sobre unidade da pressão e da temperatura. A pressão vem em `Pa`, a temperatura em `K`, a data e hora do ficheiro o *forecast time* tem intervalos de uma em uma hora desde as 0H as 12H, neste exemplo a data e a hora estão representadas da seguinte forma `201606060000 (2016/06/06 00:00)` conforme ilustrado na figura 4.9. Nestes ficheiros podemos ver ainda outros casos não muito relevantes como a precipitação, nuvens, humidade, entre outros.

A imagem 4.10, ilustra a conceção do ficheiro GRIB com o recurso à ferramenta `Panoply`.

```
105:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 0 hrs:from 201606060000
106:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 1 hrs:from 201606060000
107:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 2 hrs:from 201606060000
108:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 3 hrs:from 201606060000
109:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 4 hrs:from 201606060000
110:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 5 hrs:from 201606060000
111:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 6 hrs:from 201606060000
112:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 7 hrs:from 201606060000
113:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 8 hrs:from 201606060000
114:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 9 hrs:from 201606060000
115:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 10 hrs:from 201606060000
116:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 11 hrs:from 201606060000
117:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 12 hrs:from 201606060000
```

Figura 4.8: Dados GRIB estruturados (temperatura)

Figura 4.9: Informação dentro de um ficheiro GRIB

4.3.3 Processamento da Informação

O servidor recebe um pedido da aplicação (Cliente) com uma latitude e longitude, após receber estes dados vai processar a data e hora de quando foi realizado este pedido. O ficheiro a ser utilizado tem o nome da data e hora mais recente obtida pelo site, referido em cima. O servidor constrói caminho via esse nome para encontrar o ficheiro apropriado, que está disponível de 6 em 6 horas.

Os ficheiros GRIB podem conter vários intervalos de tempo, por exemplo, de hora em hora, de 6 em 6 horas, entre outros. Visto isto, criamos uma função para determinar o intervalo de tempo de cada ficheiro, porque é necessário saber esse intervalo para obter os valores necessários.

Existe outra função que estipula o enquadramento das horas nos intervalos, passando pelas seguintes opções, retornar o valor diretamente ou aplicar a interpolação temporal.

Analisando a figura 4.11, para retornar um valor diretamente, a hora do servidor tem de ser um valor inteiro, por exemplo, o caso das 2H, se este caso não ocorrer, sucede uma função que efetua uma interpolação temporal. Esta função como foi referido estima o intervalo, por exemplo, se a hora do servidor for 6:30 é necessário identificar o valor superior e inferior dessa hora, que poderia ser, por exemplo, 6H e 7H.

Como as horas são lidas em percentagem, foi criada uma função que transforma os argumentos passados (hora e minuto) num valor `HORA.MINUTO`, sendo os minutos em percentagem. Deste modo, se a função receber como argumento 6H e 30MIN, retorna 6.5.

Para além da interpolação temporal, foi também criada uma função para a interpolação espacial, onde o processo do cálculo dos intervalos é muito semelhante ao da interpolação temporal. Neste caso utiliza latitudes e longitudes, podendo existir a necessidade de fazer as duas interpolações sendo um processo mais complexo, denominando-se como interpolação linear.

As figuras 4.12 e 4.13 ilustram o processo da interpolação linear. Depois de efetuadas as interpolações a função retorna a pressão e temperatura.

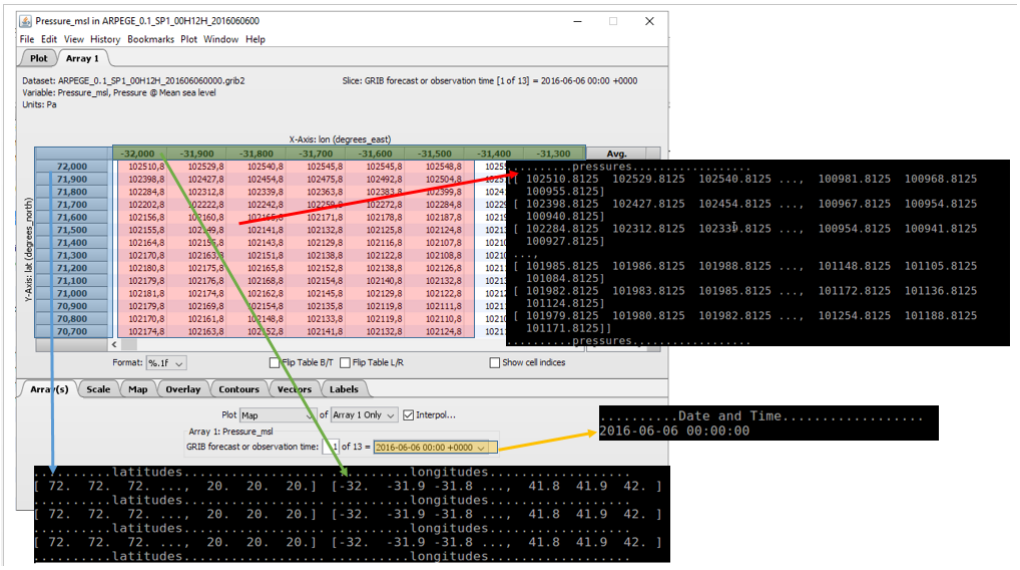

Figura 4.10: Compreensão da estrutura Grib

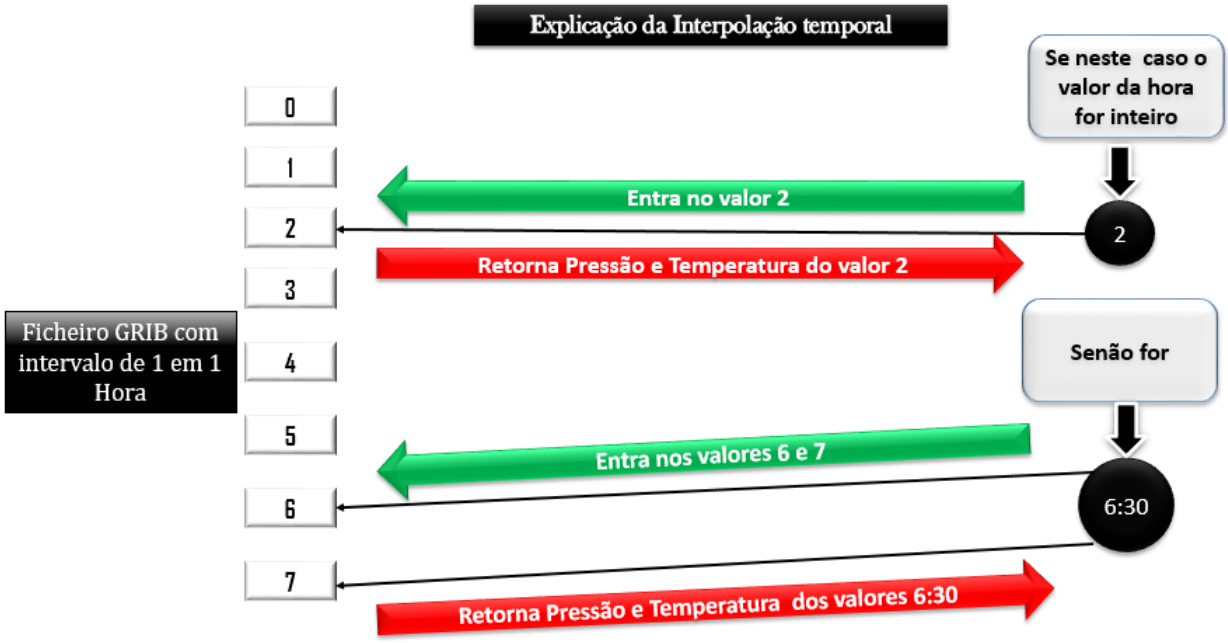

Figura 4.11: Interpolacao Temporal

4.4 Especificação do funcionamento do Cliente e sua Implementação

Quando a aplicação inicia são carregados alguns serviços. Enquanto esses serviços são carregados é mostrada uma mensagem com o texto *loading*, como se pode ver na figura 4.14. Os serviços que são carregados pela aplicação vão ser apresentados nas subsecções seguintes.

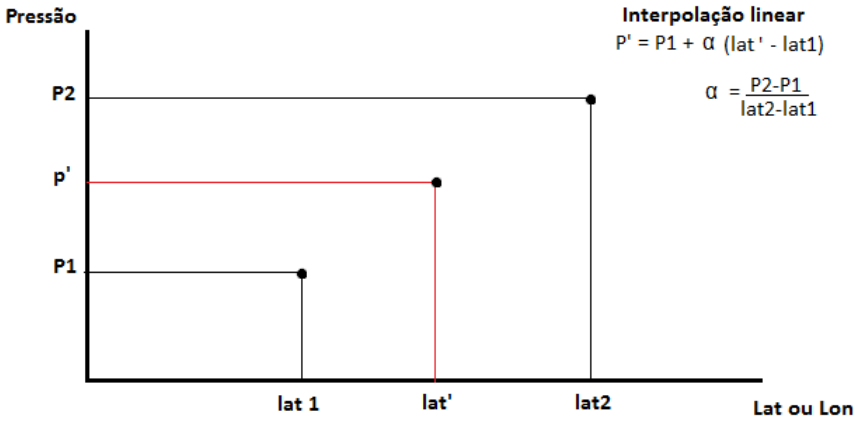

Figura 4.12: Interpolação Linear

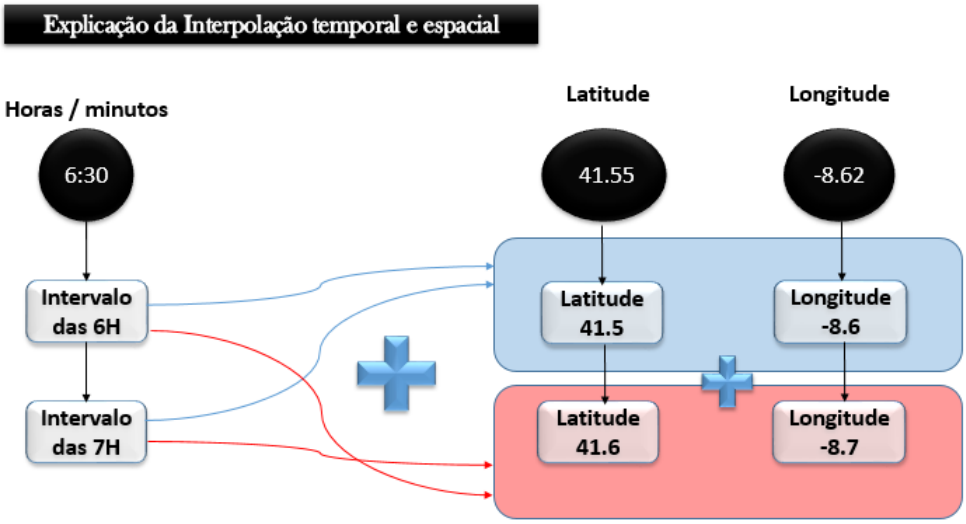

Figura 4.13: Interpolação GeoEspacial

4.4.1 Serviço GPS

O serviço GPS é invocado usando o *plugin* `cordova-plugin-geolocation` que tem como principal objetivo fornecer informações sobre a localização do dispositivo, dando à aplicação dados como a latitude e longitude. Este é suportado pelas plataformas Android, iOS, e Windows Phone [27]. Para obtermos estes dados usamos as funções, `navigator.geolocation.getCurrentPosition` e `navigator.geolocation.watchPosition`.

Atendendo ao facto de a Terra ser ligeiramente achatada nos pólos e redondo no equador, bem como as irregularidades da superfície terrestre, o presente *plugin* recorreu ao modelo Elipsoid [27] para a determinação dos valores de latitude e longitude.

Ao invocar as funções referidas anteriormente, é possível passar três parâmetros de configuração:

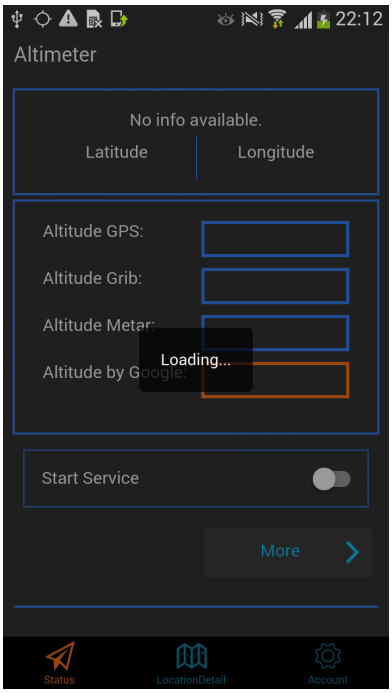

Figura 4.14: Loading

- ***Success*** - Função chamada em caso de sucesso no retorno dos dados. A estrutura de dados que é retornada por esta função é a seguinte:
  - *Latitude* - Latitude em graus.
  - *Longitude* - Longitude em graus.
  - *Altitude* - Altura da posição em metros acima do elipsóide.
  - *Accuracy* - Nível de precisão de altitude em metros.
  - *Altitude Accuracy* - Nível de precisão de latitude e longitude em metros.
  - *Heading* - Direção de deslocamento, especificado em graus sentido anti-horário em relação ao norte.
  - *Speed* - Velocidade de deslocamento atual do dispositivo, especificado em metros por segundo.
- ***Error*** - Função chamada em caso de erro no retorno dos dados. A estrutura de dados que é retornada é a mesma só que com valores a *null* ou a zero.
- ***Options*** - é utilizado para a configuração das seguintes propriedades:
  - *enableHighAccuracy*: Fornece uma aviso de que a aplicação precisa dos melhores resultados possíveis. Por padrão, o dispositivo tenta recuperar uma posição usando métodos baseados na rede. Assim ao definir a propriedade como *true* indica a *framework* para usar métodos mais precisos, como o posicionamento por satélite.

  - *timeout*: Define o máximo de tempo em milissegundos que possibilita à função, que retorna a posição, atuar. Caso o tempo limite da função seja atingido é invocada a função *Error*.

Salienta-se, que foram ainda adicionados à estrutura novos parâmetros com os nomes de `lat`, `long`, e `alt`, `isSucess` e `message`. Estes atributos são utilizados por outras funções da aplicação para arredondamentos e para o controlo de erros.

4.4.2 Serviço sensor barométrico

Este serviço é invocado pelo *plugin* `org.dartlang.phonegap.barometer`. A função que permite aceder ao sensor barométrico é a `navigator.barometer.getCurrentPressure`. Ao invocar esta função é possível passar três parâmetros:

- ***Success*** - Função chamada em caso de sucesso no retorno dos dados. A estrutura de dados retornada por esta função é:
  - *value* - Valor de pressão
  - *timestamp* - *Timestamp* do pedido em milissegundos
- ***Error*** - Função chamada em caso de erro no retorno dos dados, esta invoca a mesmos estrutura de dados a zero ou a *null*. É ainda utilizado uma condição como controlo de erros, para a verificação do sensor barométrico no dispositivo.
- ***Options*** - É utilizado para a configuração da propriedade *frequency*, que possibilita a alteração do tempo que leva a actualização da pressão.

4.4.3 Serviço Socket

Este serviço é invocado pelo *plugin* `cordova-plugin-websocket` que permite a utilização de `WebSocket` em **Android** [35]. Uma limitação deste *plugin* é a compatibilidade com as versões do **Android**. Apenas funciona nas versão superiores ao 4.0.x (*Ice Cream Sandwich*).

Para criar um `WebSocket` é necessário passar dois parâmetros, o URL e uma `string` ou um *array* de `strings`, que podem invocar as seguintes funções:

- `onopen` - Invocada na ligação do *socket*, e é aqui que deve ser enviada a mensagem pretendida para o socket através da função *send*.
- `onmessage` - É aqui que é tratada a mensagem invocada pelo *socket*.
- `onerror` - Invocada no caso de erro de ligação.
- `onclose` - Invocada no fim de ligação ao socket.

O *plugin* permite ainda configurar mais algumas propriedades, tais como:

- `origin` - Define o campo de cabeçalho do pedido.
- `maxConnectTime` - Define o tempo de espera da ligação.
- `maxTextMessageSize/maxBinaryMessageSize` - Define o tamanho da mensagem a receber do servidor.

4.4.4 Serviço METAR

A comunicação com o METAR utiliza uma função que faz uso do serviço `$http` que tem como objetivo facilitar a comunicação com um servidor HTTP. Este recebe um URL como argumento que é usado para gerar um pedido ao servidor. Após a realização do pedido, é necessário converter os dados com o formato XML para o formato JSON. Para converter esses dados foi usada a biblioteca x2js.

Os dados são estruturados e depois guardados a partir de uma função, salientando os seguintes atributos:

- *station_id* - Última observação realizada pelo METAR.
- *latitude* - Latitude fornecida pelo METAR.
- *longitude* - Longitude fornecida pelo METAR.
- *temp_c* - Temperatura fornecida pelo METAR em graus Celsius.
- *altim _in_hg* - Pressão fornecida pelo METAR em polegadas de mercúrio.
- *tempInKelvin* - Temperatura fornecida pelo METAR em Kelvin.

Adicionamos ainda à estrutura os dados fornecidos pela Google assim como dados para controlo de erros que se poderá a ver no subsecção a seguir.

4.4.5 Serviço Google

O serviço da Google é utilizado para obter os dados detalhados da localização do utilizador. Recorreu-se à função `google.maps.LatLng`, que retorna as coordenadas geográficas como latitude e longitude. As latitudes variam entre os -90 e 90 graus e as longitudes entre os -180 e 180 graus.

Para obter informação em relação as origens, destinos e modo de viagem, foi invocada a função `DistanceMatrixService`, que usa o método `DistanceMatrixService.getDistanceMatrix`. Este método inicia a solicitação do serviço matriz de distância da Google. Para a utilização da função anteriormente referida é necessário passar o seguintes dados:

- *origins* - Uma matriz contendo uma ou mais cadeias de endereços, objetos `google.maps.LatLng` ou objetos `google.maps.Place` de partida para calcular a distância e o tempo.
- *Destinations* - Uma matriz contendo uma ou mais cadeias de endereços `google.maps.LatLng`.
- *travelMode* - Modo de transporte para usar quando se calcula a rota.
- *avoidHighways* - Se for *True*, as rotas entre origens e destinos serão calculadas para evitar auto-estradas sempre que possível.
- *avoidTolls* - Se for *True*, as rotas entre os pontos serão calculadas usando rotas sem portagem, sempre que possível.

  A função retorna os seguintes dados:

- *status* - Representa o estado do *request*, caso seja válido ou não.
- *distance* - Distância total da rota em metros. O valor é expresso em segundos e como texto, sendo este formatado de acordo com o *unitSystem* especificado no pedido ou no sistema métrico.
- *duration* - Período de tempo que demora a percorrer um caminho, o valor é expresso em segundos e como texto, sendo este formatado de acordo com o *unitSystem* especificado no pedido ou no sistema métrico.

É utilizado ainda o serviço `Geocoding` que faz a conversão de endereços, como um endereço de rua em coordenadas geográficas, que podem ser utilizadas para inserir marcadores num mapa, bem como posicionar um mapa [10]. Deste modo, foi utilizado um método que através das latitudes e longitudes fornecidas, possibilita obter dados como Países, Cidades, Distritos códigos postais e entre outros.

Para a aplicação operar com os *plugins* e os serviços referidos anteriormente, foi usado o serviço `$window` que representa uma página aberta no *browser*. Todos estes pedidos são realizados de forma assíncrona, devido à utilização de um serviço denominado `$q`. Sendo este, um serviço de implementação de objetos intitulados de `promises/defered`. Para criar uma nova instância é chamada a função `$q.defer()`.

O objetivo do `defered` é expor a `promise` associada, bem como as APIs usadas para sinalizar a conclusão assim como o estado da tarefa. Este serviço pode fazer uso dos seguintes métodos:

- `resolve(value)` - Resolve a promessa enviando o valor esperado.
- `reject(reason)` - Rejeita a promessa derivada com a razão.
- `notify(value)` - Fornece atualizações sobre o estado de execução da promessa, podendo ser invocado várias vezes até que a (`promise` seja resolvida ou rejeitada.

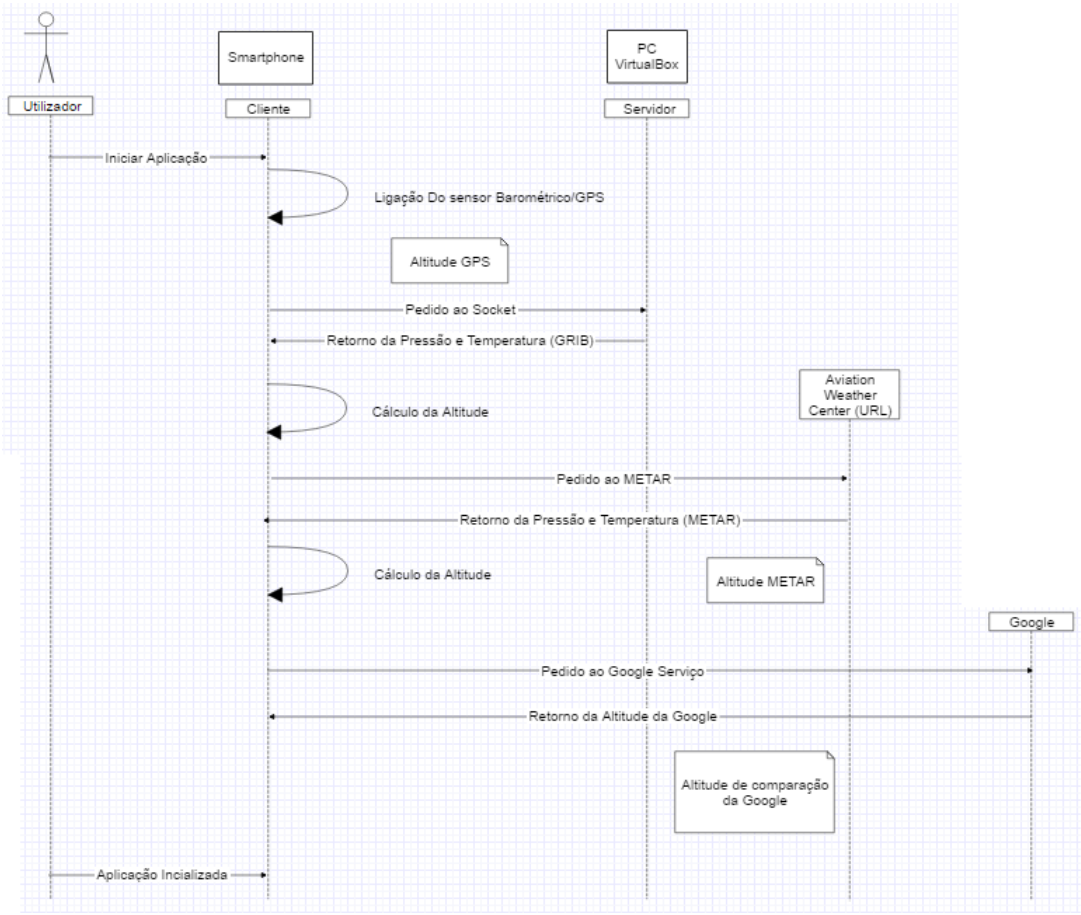

Figura 4.15: Esquema de pedidos da Altitude

A `promise` representa um valor que pode estar disponível agora, no futuro ou nunca, tendo como objetivo associar manipuladores para o eventual valor de sucesso ou fracasso de uma ação assíncrona, Permitindo assim, que métodos assíncronos retornem valores como métodos síncronos. A utilização da `promise` retorna uma promessa para o valor que terá aplicabilidade no futuro, fazendo uso dos seguintes métodos:

- `then` [(`successCallback`)], [(`errorCallback`)], [(`notifyCallback`)] - Executado independentemente de quando a *promise* foi ou será resolvida/rejeitada. Assim, logo que o resultado esteja disponível é invocada de uma forma assíncrona uma das chamadas de retorno, seja ela de êxito ou de erro.
- `finally` ((`callback`, `notifyCallback`)) - Permite observar quer a realização ou rejeição de uma `promise` sem modificar o valor final.

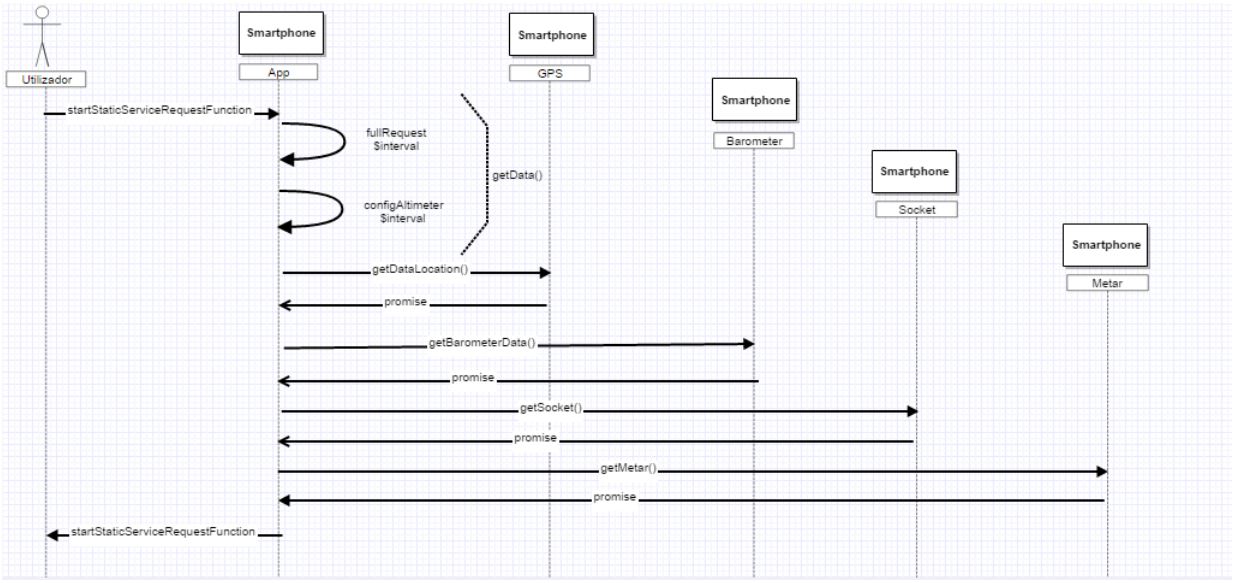

Figura 4.16: getData()

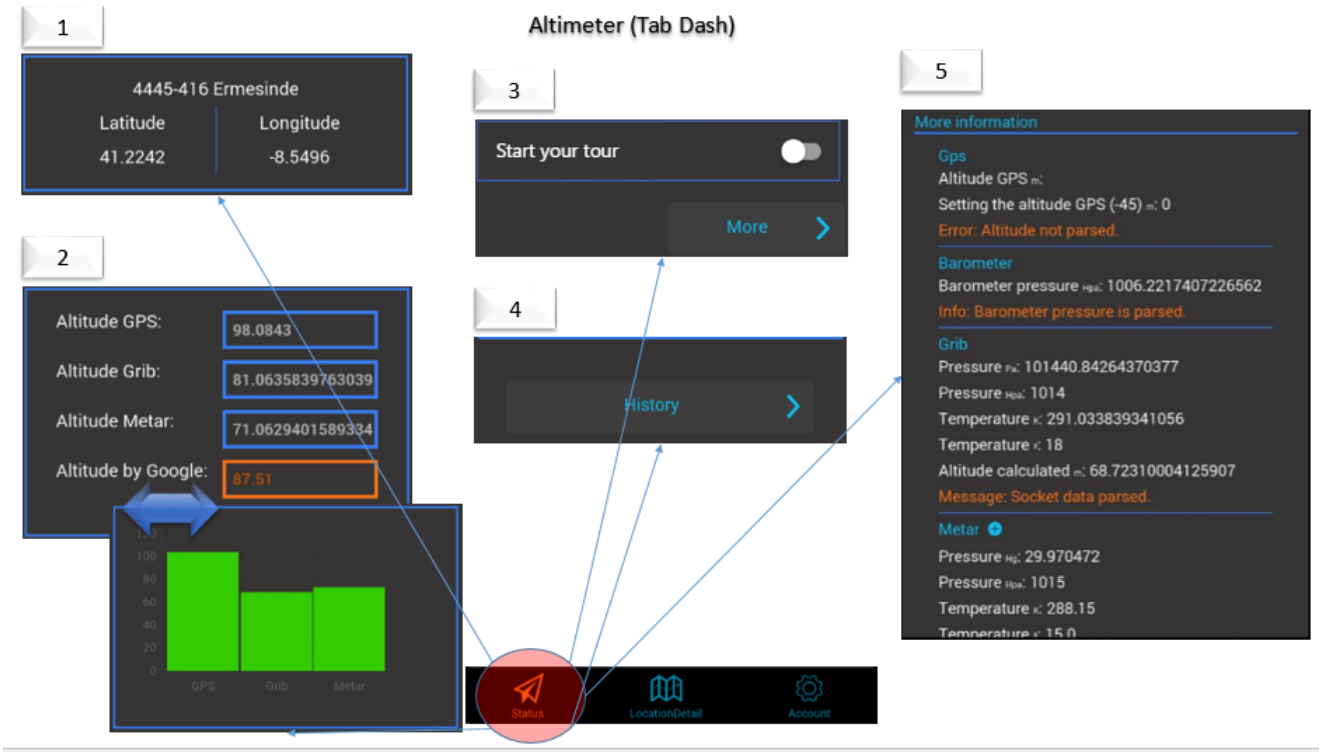

Figura 4.17: Altimetro

4.5 Apresentação da aplicação

A página inicial chama-se *Altimeter*, como podemos ver na figura 4.17. Esta é composta por várias funções e serviços, que segue de maneira geral o esquema representado na figura 4.16.

O utilizador ao iniciar a aplicação depois do *loading* estar completo, a função denominada por

`startStaticServiceRequestFunction` é invocada. Esta função faz atualização de todos os campos disponíveis do altímetro como podemos ver na figura 4.17.

1 e 2 → Ao iniciar a aplicação, a função descrita acima é invocada e os espaços são preenchidos com os dados retornados pela função `getData()`. Após a recolha dos primeiros dados, no decorrer da função `startStaticServiceRequestFunction`, são atualizados os dados provenientes do GPS e do sensor barométrico conforme o intervalo definido no `configAltimeter`. Durante este intervalo, passado um tempo vai ser feito novamente um pedido ao servidor com a função `fullRequest` que chama de novo a função `getData()` atualizando os dados. Estes dados são atualizados de acordo com as configurações dos 2 intervalos de tempo definidos pelo utilizador no *template account*, referido mais abaixo. As altitudes representadas são recebidas de diferentes formas:

- **Altitude GPS**: Recolhida através do *plugin* `cordova-plugin-geolocation`.
- **Altitude GRIB**: Recolhida após a extração dos valores da pressão e temperatura de ficheiros GRIB, enviados através da comunicação *WebSockets* entre servidor e cliente.
- **Altitude METAR**: Recolhida através de um *WebService*.
- **Altitude (Google)**: Recolhida via serviço da Google, que é usada apenas como comparação.

O utilizador da aplicação, se deslizar o dedo (slide) no local onde as altitudes estão apresentadas, consegue ver estes valores através de um gráfico.

3 → Possui também um botão `ON/OFF` com o nome de *Start your tour*, designada para o movimento do utilizador, ou seja para a elaboração de passeios.

4 → Possui também um botão com o nome de *History*, designada para guardar serviços realizados pelo utilizador.

5 → Para o utilizador perceber os dados que estão ou não a ser recebidos.

O *Start Service* ilustrado na figura 4.18 é um serviço concebido para guardar dados sempre que o utilizador estiver em andamento, por exemplo, numa caminhada. Este guarda vários tipos de dados como pressão, temperatura, rua, entre outros. De acordo com a figura 4.19, o *Start Service* passa por alguns passos:

1 → O utilizador liga o serviço *Start your tour* que invoca uma função `startStop()`. Após a ligação do serviço, é invocado o `getData()`. No momento que este é iniciado, são guardadados dados no `localStorage` com a utilização do serviço `servicePlaces` e a função `savePlace` com a *key service*. Estes dados estão continuamente a ser gravados neste serviço de acordo com o intervalo de tempo definido.

O utilizador ao desligar o serviço `servicePlaces`, é invocada a função `finishService()`

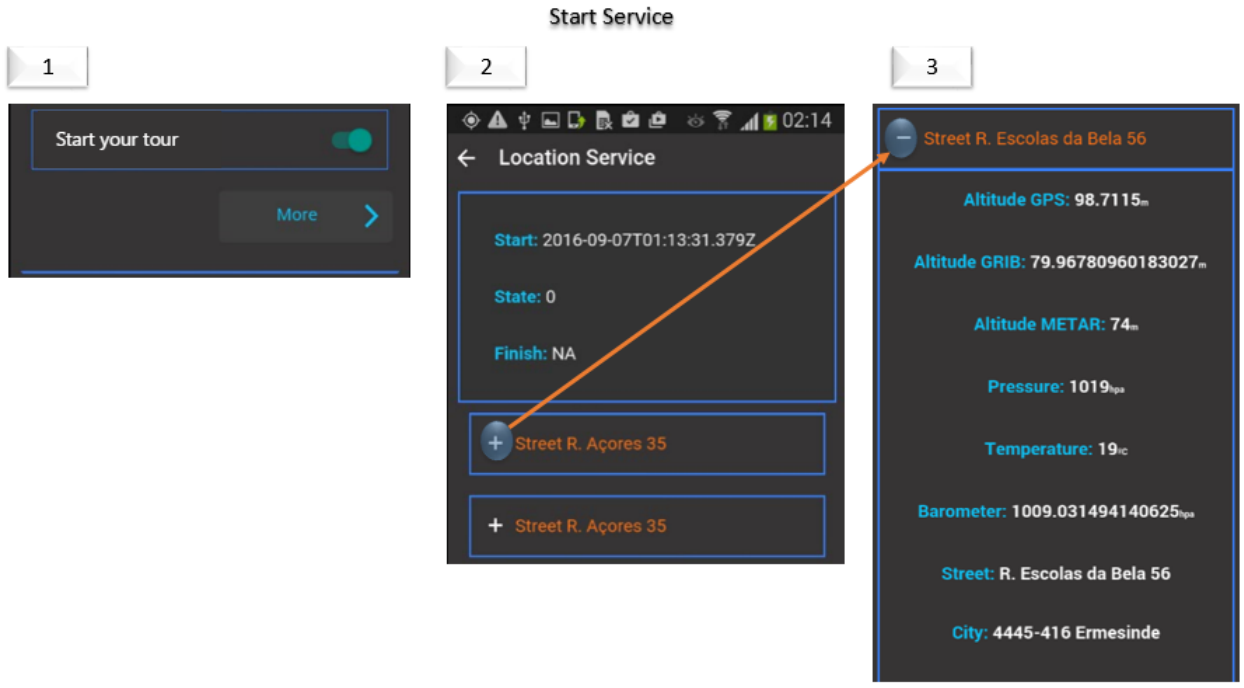

Figura 4.18: Serviço de Passeio

Figura 4.19: Esquema do Serviço Start Service

que adiciona a *key services* do `localStorage` que foi guardada durante a execução do serviço *Start your tour*. Removendo do `localStorage` a *key service*.

2 → No decorrer do passeio ou corrida, com um *click* no botão `More` é possível verificar a hora e o dia que foi iniciado o pedido, e sempre que o serviço faz um pedido novo, este guarda o nome da rua em que o utilizador está.

3 → Ao clicar no símbolo + atrás do nome da rua, mostra mais informações das ruas que passou.

No *template* inicial mostra os serviços guardados do *Start your tour* fazendo assim um histórico, como podemos ver na figura 4.20, que tem as seguintes representações:

1 → O utilizador clica no botão `History`, que abre um *template* com todos os serviços ou passeios.

2 → Neste *template* o utilizador pode identificar e escolher o passeio que realizou no passado, verificando pela data/hora de inicio ou de fim desse serviço. Possui ainda duas opções, apagar um dos serviços antigos com o botão *delete*, ou clicar no botão *more*.

3 → Ao clicar no botão *more* é possível ver mais detalhadamente os passeios que percorreu, podendo obter informações como as altitudes por onde passou e entre outras.

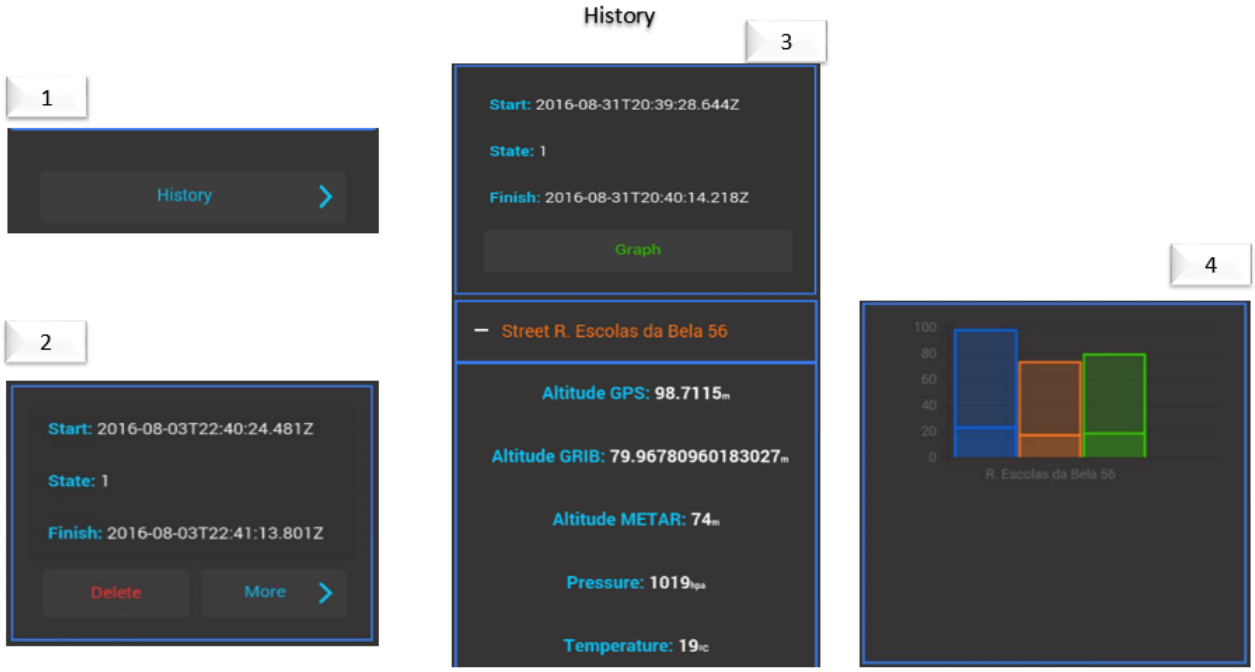

Figura 4.20: Historial

4 → É possível ainda, através do *template* anterior carregar no botão *Graph* e ver as altitudes em que utilizador esteve em forma de gráfico.

O *template Location Detail* é utilizado para descobrir informações da pressão e da temperatura através do GRIB e do METAR de qualquer localização. Atualmente está definido para a localização do utilizador, como se pode ver na figura 4.21. Este *template* passa pelas seguintes etapas e funções:

1 → Após a inserção de uma latitude e longitude nas caixas de texto, seguidamente de um *click* no botão *track* localiza o local inserido através de GPS. O *radial distance* serve para localizar um METAR mais próximo dependendo do alcance que é facultado. O máximo alcance que lhe pode ser atribuído é de 800 quilómetros.

2 → Os dados, como podemos ver nesta imagem, são recolhidos do ficheiro GRIB mais atualizado e do METAR do local mais próximo.

No *template Location Detail* existe um botão que fornece as seguintes informações:

1 → Ao clicar no botão *more* abre um *template* com o nome de *map METAR*, como se pode ver na figura 4.22.

2→ O *template* ao abrir, possui informações sobre o METAR mais próximo de acordo com a localização que foi fornecida, e o *radial distance* estabelecido. Com este *template* é possível saber qual a distância do METAR mais próximo em linha reta assim como a distância definida pelo `Geocoding`.

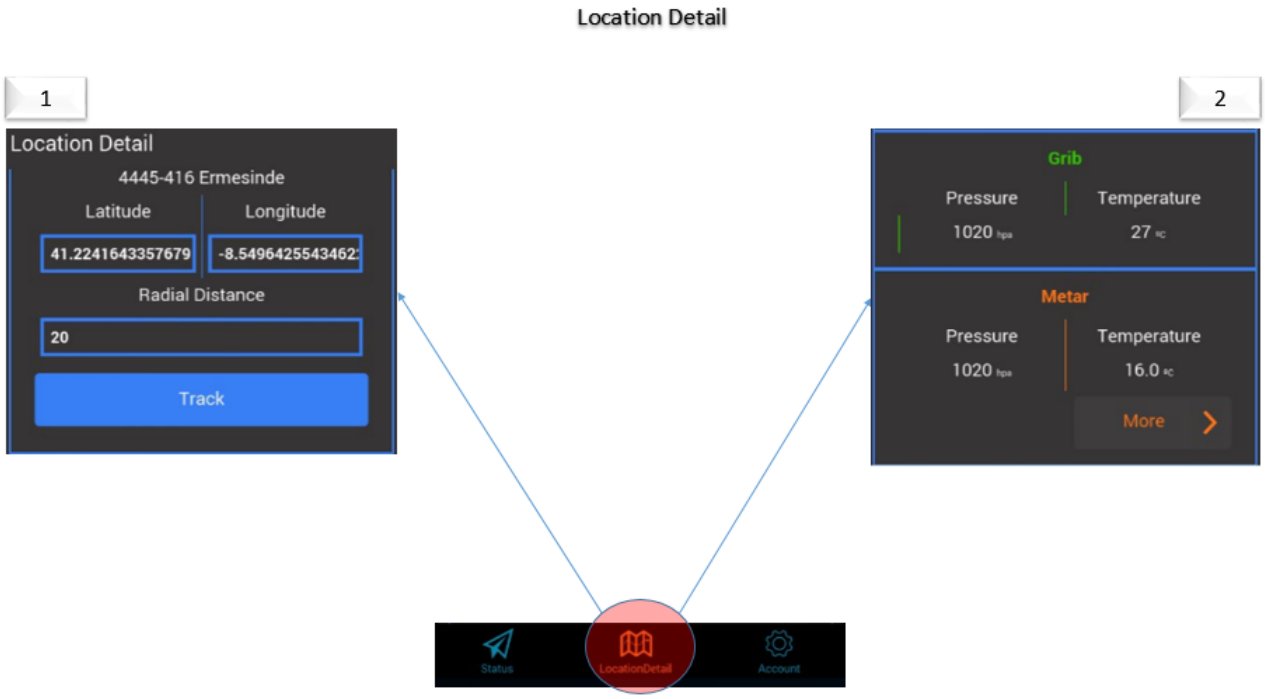

Figura 4.21: Localização

3→ Neste *template* é possível verificar a distância em linha reta no mapa desde o local estabelecido até ao METAR mais próximo. No *template account* são consideradas as definições do altímetro, e apresentam as funcionalidades representadas na figura 4.23:

1 → Como se pode verificar este *template* usa sempre a mesma porta e endereço IP privado ou público. O nome que está a ser utilizado corresponde ao endereço público da rede do servidor.

2 → O *Barometer Corretion Value (hPa)* permite adicionar ou subtrair um valor em hectopascais, para a calibrar o barómetro .

O *Time of route actualization (ms)* permite mudar a duração da atualização da informação do *Start your tour* em milissegundos.

O *(Altimeter Actualization (ms))* permite mudar a duração dos pedidos realizados ao GPS e ao sensor barométrico em milissegundos.

O *(Time of all Requests)* permite mudar a duração dos pedidos ao GPS, `Socket`, Sensor Barométrico e METAR.

O *(Change the Radial Distance (kms))* permite aumentar o alcance em quilómetros para poder localizar os METARs mais próximos, permitindo obter assim informações dos mesmos.

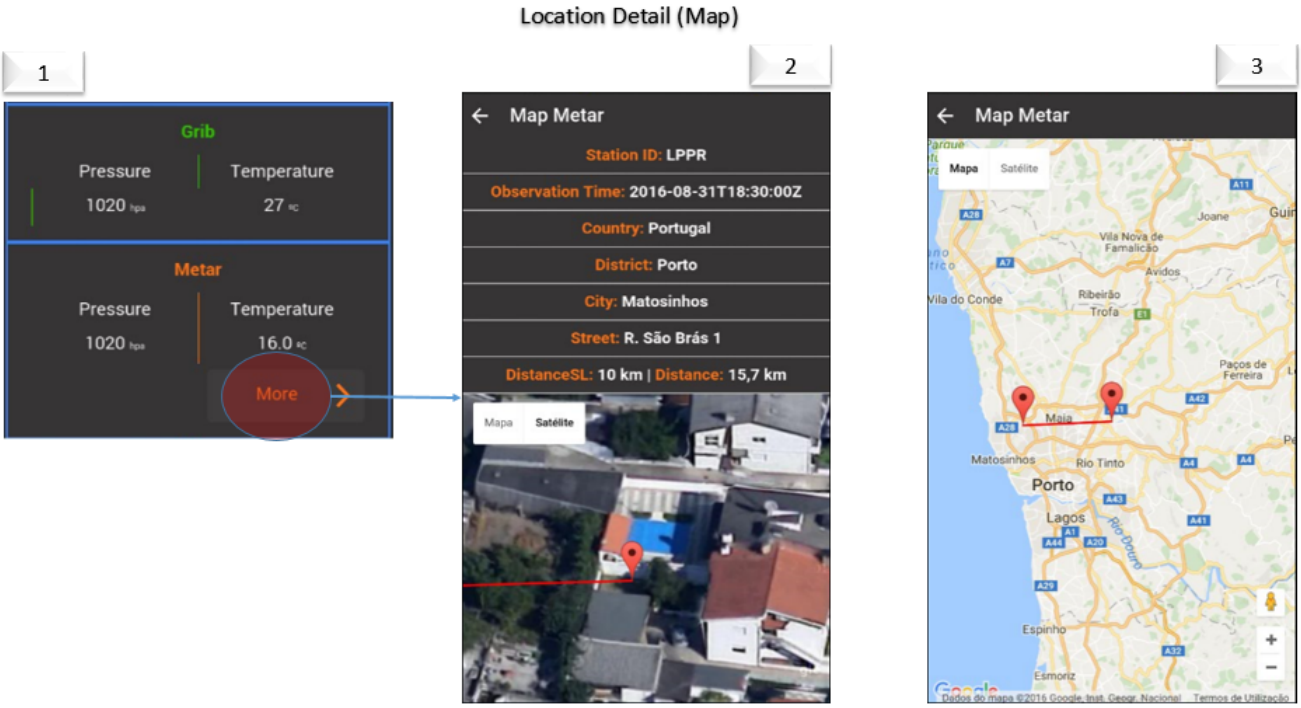

Figura 4.22: Mapa

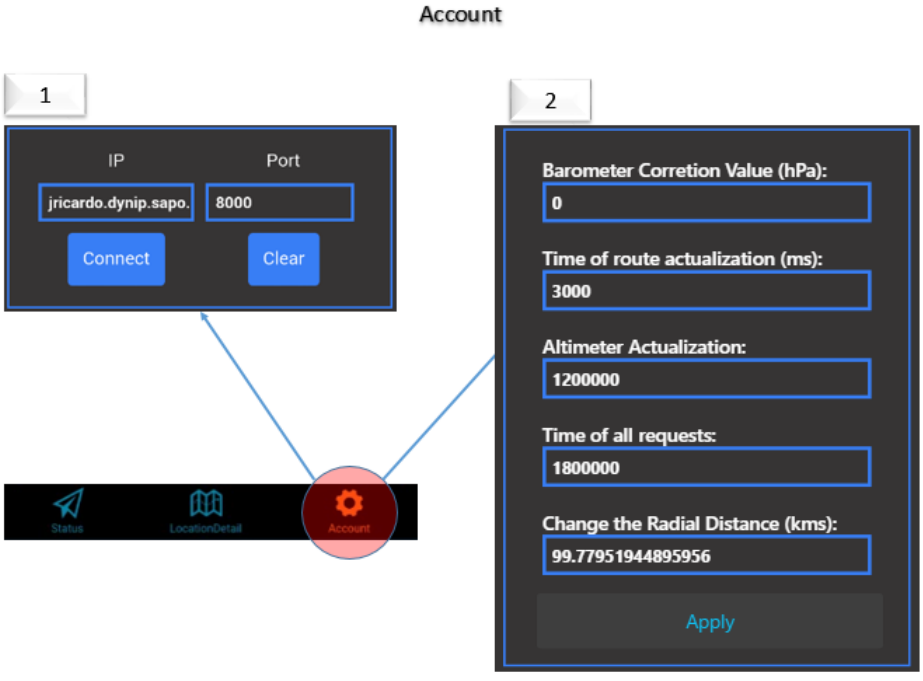

Figura 4.23: Definições

Capítulo 5

Avaliação

Neste capítulo vão ser apresentados os resultados dos testes realizados à aplicação. Para esse efeito, foi necessário a construção de um servidor de modo a guardar os dados provenientes da aplicação (visto que esta permite apenas a visualização dos mesmos). Assim para a construção do servidor foi utilizado e instalado o **XAMPP**, dado que este possui as tecnologias necessárias e mais comuns para este efeito tais como o Hypertext Preprocessor (PHP), Apache entre outros. Foi então desenvolvido uma *Representational State Transfer (REST) API*, que comunica com a aplicação (Cliente), com o objetivo de guardar os dados recolhidos pelo altímetro [20].

O esquema apresentado na figura 5.1 representa o modo como é feita a comunicação entre o cliente e a REST API. O cliente comunica com a REST API desenvolvida em *PHP* através de um pedido *HTTP*. Esta foi desenvolvida utilizando o *Slim framework* (uma *microframework* PHP que ajuda a escrever aplicações web simples para APIs). A aplicação ao chamar a função `doCrossDomainGet` invoca um *URL* através de uma HTTP *POST*. Esta função envia no corpo da mensagem um *JSON*, no qual a sua estrutura contém dados como altitudes, pressão e temperatura. Os dados necessários e utilizados dessa estrutura para os testes, são enviados para dois ficheiros `csv` que são: o `data.csv` e `pressureTemperature.csv`. O primeiro é composto pelos dados Data/Hora, e as altitudes do *GPS*, *METAR* e *GRIB*. O segundo ficheiro, era composto pelos valores da pressão medida pelo sensor barométrico do *smartphone*, e pelas temperaturas provenientes das estações terrestres.

Através dos dados recolhidos e guardados nestes ficheiros foram realizados os gráficos **dados do altímetro**, **variação de erros**, **variação da pressão atmosférica** e **variação das temperaturas**

5.1 Testes e Resultados

A nossa experiência consistiu em colocar um *smartphone* com a aplicação instalada, num sítio onde fosse possível obter o sinal do GPS. O dispositivo permaneceu imóvel durante um período de 3 dias num local com um valor real de altitude de 92 m [51].

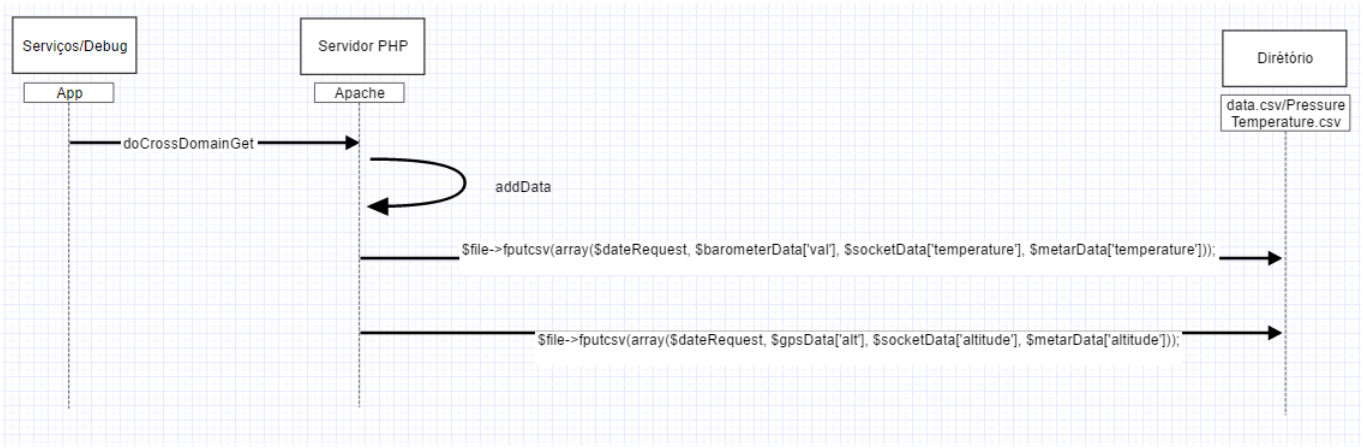

Figura 5.1: Esquema de como os dados foram Guardados

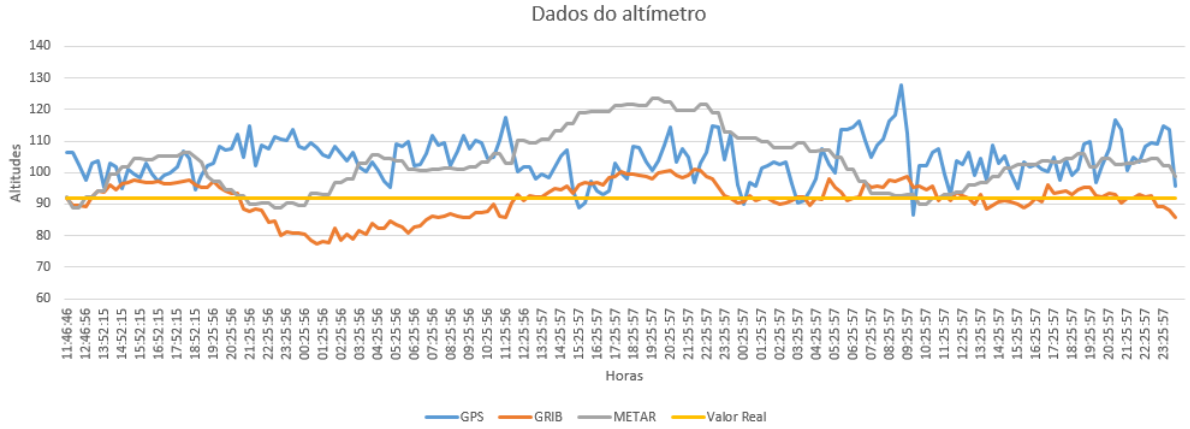

Figura 5.2: Dados lidos pelo Altímetro

Como podemos observar no gráfico 5.2, todas as altitudes sofreram alterações, o que nos leva a concluir a falta de estabilidade em todos os métodos avaliados. Supõe-se que estas acontecem devido às diferenças de temperatura e pressão atmosférica. De facto, se analisarmos os valores de pressão e temperatura durante o período da recolha (gráficos 5.3 e 5.4), podemos observar que, com o passar do tempo, os valores correspondentes vão variando, afetando as medições da altitude.

Depois de observadas as diferenças de altitude de cada um dos métodos no gráfico 5.2, foi feita uma análise para comparar os valores de erros dos mesmos. O gráfico 5.5 representa a variação de erro dos três métodos no decorrer do tempo da experiência. Após o estudo realizado verificou-se que as estações terrestres têm um erro menor equiparando com GPS. De facto, ao analisar a média do valor absoluto do erro, conclui-se que o METAR apresenta o valor mais baixo, o que significa que é uma das opções mais confiáveis com uma média de erro por dia de 4.3 metros. No que consta ao GRIB e ao GPS tiveram uma diferença de erro maior, sendo que o GRIB chegou a ter uma diferença de 10 metros e o GPS de 12 metros.

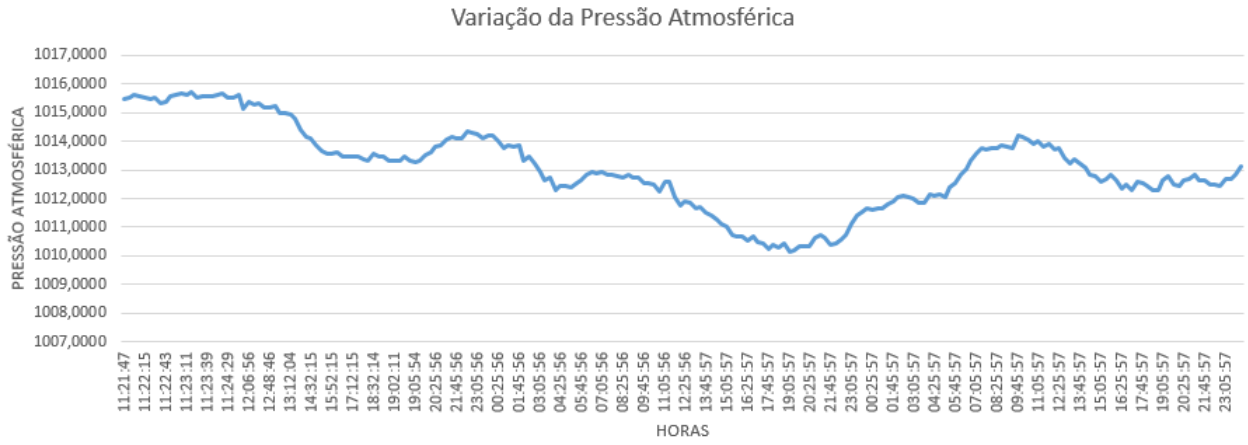

Figura 5.3: Pressão do sensor Barométrico

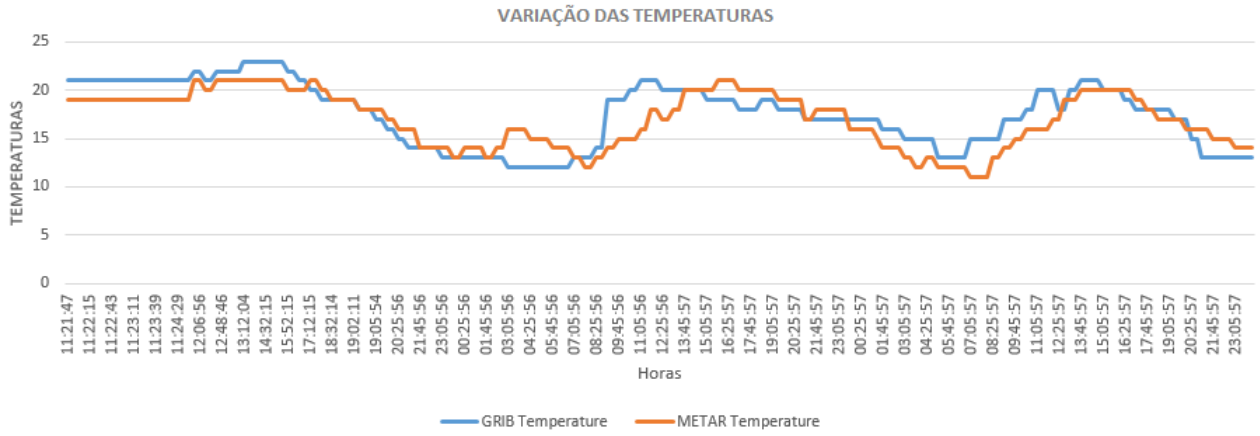

Figura 5.4: Temperatura GRIB/METAR

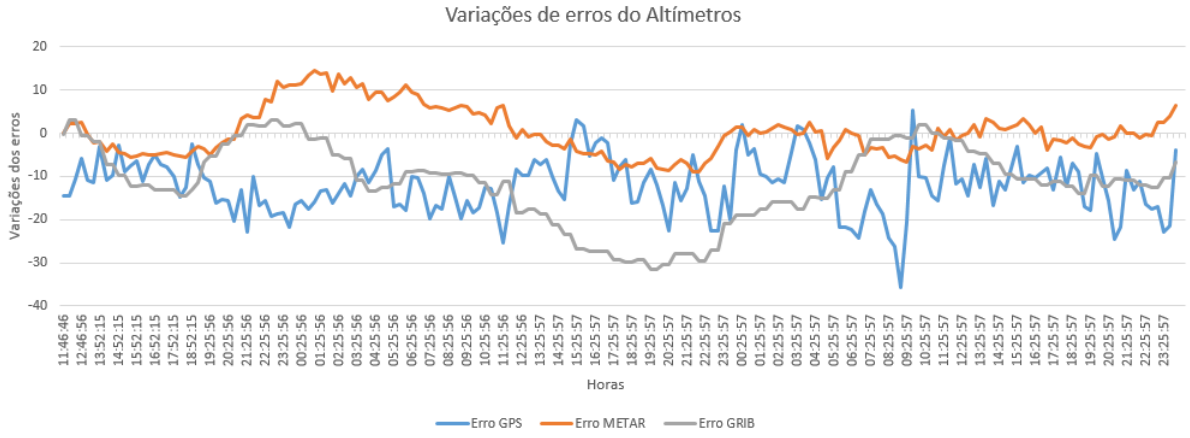

Figura 5.5: Erros Altímetro

Capítulo 6

Conclusões

No decorrer desta dissertação de mestrado, o objetivo principal consistia no desenvolvimento de uma aplicação para dispositivos móveis, que permitisse estimar a altitude do utilizador através de vários mecanismos. Através de um conjunto de métodos para obter a altitude, foi possível atingir esse objetivo. A altitude foi obtida com o auxílio de ficheiros *GRIB*, *METARs* e *GPS*, com o auxílio do sensor barométrico incorporado no *smartphone*. Pode-se dizer que a aplicação desenvolvida é eficiente e multi-funcional, ou seja, a aplicação pode ser executado nos três principais sistemas operativos para dispositivos móveis (*Android*, iPhone OS (iOS) e Windows Phone).

Foram obtidos ainda grandes conhecimentos de técnicas de determinação da altitude e de localização via GPS. Desenvolveu-se ainda técnicas que aperfeiçoaram o cálculo de altitude por intermédio de estações barométricas terrestres e do sensor barométrico.

Através dos resultados dos testes realizados, é possível concluir que, com o auxílio do sensor barométrico e da ajuda dos dados recolhidos pelas observações dos METARs e das previsões dos ficheiros GRIB, estes têm uma melhor altitude que a proveniente do GPS devido as suas constantes variações. Podemos concluir também que a pressão e a temperatura são fatores importantes na determinação da altitude precisa.

6.1 Aspetos a melhorar

É possível melhorar a aplicação, com um melhor planeamento do seu desenvolvimento. No entanto como aspetos a melhorar existem estas possibilidades:

- Melhorar a desempenho da aplicação porque é lenta a iniciar, devido à quantidade de pedidos que são realizados.
- Aumentar o controlo de erros no que consta a avisos. Por exemplo, alertar o utilizar se não tiver o GPS ligado, porque deste modo a aplicação não vai funcionar como é esperado.

6.2 Trabalho Futuro

Para trabalho futuro existem muitas possibilidades como por exemplo:

- Adicionar previsões meteorológicas, como a possibilidade de chuva para o dia seguinte, ventos fortes, ou neve, através de adição de *plugins* já existentes.
- Ligar com redes sociais como por exemplo, Facebook, Twitter, Instagram, partilhando por exemplo a maior altitude que o utilizador já esteve ou está no momento.
- Possibilidade de integrar uma opção fotográfica, que identificasse o local mais alto em que utilizador está ou quer atingir (ao tirar uma foto no monte Evereste, através do valor da altitude medido, identificar a montanha). Como também identificar altitudes de monumentos históricos, via foto (tirar fotografia à Torre Eiffel e identificar que ela tenha 324 metros).
- Incluir o modo *offline*.
- Incorporar conversor para todas unidades dos valores da altitude, temperatura e pressão atmosférica.
- Registar por meio de um gráfico a altitude mais baixa e mais alta que o utilizador presenciou.
- Incorporação de bússola.
- Calibração automática.

Bibliografia

[1] abdmob. x2js. https://github.com/abdmob. Acedido online em: 2016-06-10.

[2] C Donald Ahrens. *Essentials of meteorology: an invitation to the atmosphere*. Cengage Learning, 2011.

[3] José Milton Arana. O uso do gps na determinação de altitudes ortométricas. In *Congresso Brasileiro de Cadastro Técnico Multifinalitário*, volume 6, 2004.

[4] Dmitry Bershadsky, Louis Dressel, and Eric Johnson. Situational and terrain awareness and warning system implementation on android smartphone for manned aviation applications. *AIAA SciTech, National Harbor, MD, Jan*, pages 13–17, 2014.

[5] Tim Bray. The json data interchange format. *draft-ietf-json-rfc4627bis-10 (work in progress)*, 2013.

[6] Nicko Brennan. Pocketgrib, . Acedido online em: 2016-04-13.

[7] Nicko Brennan. Pocketgrib, . Acedido online em: 2016-04-13.

[8] João Carlos Cardoso. Leituras de altitude a partir de um recetor gps. https://landlousa.wordpress.com/2011/04/23/leituras-de-altitude-a-partir-de-um-receptor-gps-geometria-dos-satelites/, Abril 2011. Acedido em: 2016-06-02.

[9] Dora Catarina. O sistema gps e o funcionamento dos seus segmentos. http://perceberomundo.blogs.sapo.pt/742.html, 2006. Acedido em: 2016-07-06.

[10] Google Company. What is geocoding? https://developers.google.com/maps/documentation/geocoding/intro, . Acedido em: 2016-07-06.

[11] Google Company. $http. https://docs.angularjs.org/api/ng/service/\protect\T1\textdollarhttp, . Acedido em: 2016-06-02.

[12] Tutorials Point Company. Node js tutorials point simplye as learning. . Acedido em: 2016-06-20.

[13] Tutorials Point Company. Web socket tutorials point simplye as learning. . Acedido em: 2016-05-02.

[14] Carlos Correia. Sistema gps–introdução. *Sistemas de Telecomunicações II, FEUP, Porto*, 2003.

[15] Jorge Luiz Barbosa da Silva. Nivelamento geométrico. *Topografia 1*, 2003.

[16] Peter B. de Selding. Europe positioned to order additional galileo satellites soon. http://spacenews.com/europe-positioned-order-additional-galileo-satellites-soon/. Acedido em: 2016-03-9.

[17] NumPy Developers. Numpy. http://www.numpy.org/index.html. Acedido em: 2015-11-20.

[18] Rui Dilão. Gps. http://sd.ist.utl.pt/NonLinear_Dynamics_Group/Awareness_files/gps.pdf. Acedido em: 2016-05-10.

[19] Daniel Dimov. True altitude calculations. page 5, February 2013.

[20] Dalibor D. Dvorski. Installing, configuring, and developing with xampp. page 10, March 2007.

[21] Mário Quadro e Ruy de Sa. VisÃo geral dos aplicativos para meteorologia. page 12. Instituto Federal Santa Catarina, February 2010.

[22] Helder Ferreira. *Sistemas de Instrumentação*. ECEF, February 2009. Acedido em: 2016-05-23.

[23] Heinz Burda Filho. altímetro - nocões básicas. http://hangardoheinz.blogspot.pt/2010/06/altimetro-nocoes-basicas.html. Acedido em: 2016-02-16.

[24] Mariano Furriel. Websockets vs. long polling. http://fdvsolutions.com/blog/websockets-vs-long-polling/. Acedido em: 2016-05-03.

[25] Emerson Galvani. Pressão atmosférica. In *Climatologia I*, page 37. Universidade de Sao Paulo, Departamento de Geografia, 2014.

[26] Rohit Ghatol and Yogesh Patel. *Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5*. Apress, 2012.

[27] Steve Gill. cordova-plugin-geolocation. https://github.com/apache/cordova-plugin-geolocation. Acedido em: 2016-06-16.

[28] Tiago Gonçalves. Altimetria. Online. Acedido em: 2016-04-13.

[29] Como hacer para. Las mejores aplicaciones para viajeros. http://comohacerpara.com/las-mejores-aplicaciones-para-viajeros_12002s.html.

[30] Cátia Homem. Viagens com gps. http://fisicaequimica11.blogspot.pt/2008/09/viagens-com-gps.html. Acedido em: 2016-05-05.

[31] AWS-SAA J Miguel Mendez. Ionic/cordova 4 month opinion. http://fisicaequimica11.blogspot.pt/2008/09/viagens-com-gps.html. Acedido em: 2016-05-23.

[32] Andrea Ritter Jelinek. Altimetria. In *Topografia 1*.

[33] J. Moura Ramos Joaquim and Leitão Luis. A atmosfera da terra. *instrumetos de laboratorio e cientificos lda*, page 11, 2001.

[34] Elliott Kaplan and Christopher Hegarty. *Understanding GPS: principles and applications*. Artech house, 2005.

[35] knowledgecode. cordova-plugin-websocket. https://github.com/knowledgecode.

[36] Guangwen Liu, Khan Muhammad Asif Hossain, Masayuki Iwai, Masaki Ito, Yoshito Tobe, Kaoru Sezaki, and Dunstan Matekenya. Beyond horizontal location context: measuring elevation using smartphone's barometer. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 459–468. ACM, 2014.

[37] Juliana Marcico. Os elementos do clima. http://geograficmarcico.blogspot.pt/2016/09/os-elementos-do-clima.html. Acedido em: 2016-04-10.

[38] Elaine Martins. Como funciona o gps. https://www.tecmundo.com.br/gps/2562-como-funciona-o-gps-.htm. Acedido em: 2016-04-02.

[39] Microsoft. Sockets for windows phone 8. https://msdn.microsoft.com/en-us/library/windows/apps/hh202874(v=vs.105).aspx. Acedido em: 2016-05-21.

[40] Shahram Najm. Development section. page 10, March 2007.

[41] Shahram Najm. Slide 1grib api, feb 2015 ecmwfslide 1grib api. ECMWF, February 2015.

[42] Mozilla Developer Network and individual contributors. Xmlhttprequest. https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest, . Acedido em: 2016-06-02.

[43] Mozilla Developer Network and individual contributors. Promise. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise, . Acedido em: 2016-06-01.

[44] Rodolfo F. Alves Pen. Pressão atmosférica. http://mundoeducacao.bol.uol.com.br/geografia/pressao-atmosferica.htm, 2016. Acedido em: 2015-12-29.

[45] Gabriel Philipe. Como funciona o gps. https://www.oficinadanet.com.br/post/12406-como-funciona-o-gps. Acedido em: 2016-03-28.

[46] Grillo Rafael. Introducão ao ionic framework. https://tableless.com.br/introducao-ao-ionic-framework/, February 2015. Acedido em: 2016-05-05.

[47] Frank Singleton. Understanding, getting and using grib weather files. http://weather.mailasail.com/Franks-Weather/Grib-Files-Getting-And-Using. Acedido em: 2015-11-10.

[48] Marco Dias Tiago Gonçalves. Metar speci taf. page 6, 2005.

[49] Paula Tomaz. Coordenadas geográficas. https://afiaagvv.blogspot.pt/2008/02/coordenadas-geogrficas.html, . Acedido em: 2016-04-01.

[50] Paula Tomaz. Coordenadas geográficas. https://geografiaagvv.blogspot.pt/2008/02/coordenadas-geogrficas.html, . Acedido em: 2016-04-01.

[51] topographic-map company. Portugal. http://pt-pt.topographic-map.com/places/Portugal-4004875/. Acedido em: 2016-07-01.

[52] what-when how. How gps works. http://what-when-how.com/molecular-biology/s-adenosyl-l-methionine-molecular-biology/. Acedido em: 2016-02-12.

[53] Jeffrey Whitaker. Module pygrib. http://jswhit.github.io/pygrib/docs/, December 2014. Acedido em: 2015-11-25.

[54] Wikiwand. Código aeroportuário icao. http://geograficmarcico.blogspot.pt/2016/09/os-elementos-do-clima.html. Acedido em: 2016-04-29.

Apêndice A

Acrónimos

API Aplication Programming Interface

CSS Cascading Style Sheets

DEM Digital Elevation Model

DNS Domain Name System

DOM Document Object Model

ECEF Earth-Centered Earth-Fixed

ECMWF European Centre for Medium-Range Weather Forecasts

FTP File Transfer Protocol

GDS Grid Descrição Seção

GFS Global Forecast System

GNSS Global Navigation Satellite System

GPS Global Positioning System

GRIB General Regularly-distributed Information in Binary form

HDF Hierarchical Data Format

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

ICAO International Civil Aviation Organization

ID Identificação Digital

IDE Integrated Development Environment

iOS iPhone OS

ISA International Standard Atmosphere

ISP Internet Service Provider

IP Internet Protocol

JSON JavaScript Object Notation

JPEG Joint Photographic Experts Group

JScript JavaScript

METAR METeorological Aerodrome Report

METARs METeorological Aerodrome Report

MOB Man Overboard

MVC Model View Controller

MSL Mean Sea Level

NAVSTAR NAVigation System with Time And Ranging Global Positioning System

netCDF Network Common Data Form

NOAA National Oceanic and Atmospheric Administration

OACI International Civil Aviation Organization

PC Personal Computer

PDF Portable Document Format

PDS Product Definition Section

PHP Hypertext Preprocessor

PNG Portable Network Graphics

REST Representational State Transfer

SDK Software development kit

SPECI Sample of Aviation Special Weather Report in Code

TCP Transmission Control Protocol

TIFF Tagged Image File Format

UDP User Datagram Protocol

URL Uniform Resource Locator

UTC Universal Time Coordinated

UI User Interface

XML eXtensible Markup Language

WS windshear